

Podręcznik FreeBSD

Projekt Dokumentacji FreeBSD

Podręcznik FreeBSD

Projekt Dokumentacji FreeBSD

Data wydania Luty 1999

Copyright © 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006 Projekt Dokumentacji FreeBSD

Witamy w świecie FreeBSD! Zadaniem niniejszego podręcznika jest opisanie procesu instalacji i czynności związanych z codziennym użytkowaniem systemu FreeBSD w wersji 8.3-RELEASE oraz 9.1-RELEASE. Prace nad tym podręcznikiem trwają *cały czas*. Stanowi on dzieło wielu osób z całego świata. Tym nie mniej mamy świadomość, iż wiele rozdziałów wciąż nie zostało napisanych, a niektóre spośród istniejących wymagają aktualizacji. Jeśli jesteś zainteresowany pomocą w rozwoju projektu wyślij email na adres listy dyskusyjnej projektu dokumentacji FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc>). Najnowsza wersja angielszczyzna niniejszego dokumentu jest zawsze dostępna na stronie domowej FreeBSD (<http://www.FreeBSD.org/>) (wersje wcześniejsze dostępne są pod adresem <http://docs.FreeBSD.org/doc/>). Podręcznik dostępny jest również w innych formatach dokumentów oraz w postaci skompresowanej z serwera FTP Projektu FreeBSD (<ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/>) bądź jednego z wielu serwerów lustrzanych. Dla osób zainteresowanych, drukowaną wersję podręcznika (język ang.) można nabyć wprost z witryny FreeBSD Mall (<http://www.freebsdmail.com/>). Dostępne jest również przeszukiwanie podręcznika (<http://www.FreeBSD.org/search/index.html>).

Redistribution and use in source (SGML DocBook) and 'compiled' forms (SGML, HTML, PDF, PostScript, RTF and so forth) with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code (SGML DocBook) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.
2. Redistributions in compiled form (transformed to other DTDs, converted to PDF, PostScript, RTF and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

WAŻNE: THIS DOCUMENTATION IS PROVIDED BY THE FREEBSD DOCUMENTATION PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FREEBSD DOCUMENTATION PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

FreeBSD is a registered trademark of the FreeBSD Foundation.

3Com and HomeConnect are registered trademarks of 3Com Corporation.

3ware and Escalade are registered trademarks of 3ware Inc.

ARM is a registered trademark of ARM Limited.

Adaptec is a registered trademark of Adaptec, Inc.

Adobe, Acrobat, Acrobat Reader, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Apple, AirPort, FireWire, Mac, Macintosh, Mac OS, Quicktime, and TrueType are trademarks of Apple Computer, Inc., registered in the United States and other countries.

Corel and WordPerfect are trademarks or registered trademarks of Corel Corporation and/or its subsidiaries in Canada, the United States and/or other countries.

Sound Blaster is a trademark of Creative Technology Ltd. in the United States and/or other countries.

CVSup is a registered trademark of John D. Polstra.

Heidelberg, Helvetica, Palatino, and Times Roman are either registered trademarks or trademarks of Heidelberger Druckmaschinen AG in the U.S. and other countries.

IBM, AIX, EtherJet, Netfinity, OS/2, PowerPC, PS/2, S/390, and ThinkPad are trademarks of International Business Machines Corporation in the United States, other countries, or both.

IEEE, POSIX, and 802 are registered trademarks of Institute of Electrical and Electronics Engineers, Inc. in the United States.

Intel, Celeron, EtherExpress, i386, i486, Itanium, Pentium, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intuit and Quicken are registered trademarks and/or registered service marks of Intuit Inc., or one of its subsidiaries, in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

LSI Logic, AcceleRAID, eXtremeRAID, MegaRAID and Mylex are trademarks or registered trademarks of LSI Logic Corp.

M-Systems and DiskOnChip are trademarks or registered trademarks of M-Systems Flash Disk Pioneers, Ltd.

Macromedia, Flash, and Shockwave are trademarks or registered trademarks of Macromedia, Inc. in the United States and/or other countries.

Microsoft, IntelliMouse, MS-DOS, Outlook, Windows, Windows Media and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Netscape and the Netscape Navigator are registered trademarks of Netscape Communications Corporation in the U.S. and other countries.

GateD and NextHop are registered and unregistered trademarks of NextHop in the U.S. and other countries.

Motif, OSF/1, and UNIX are registered trademarks and IT DialTone and The Open Group are trademarks of The Open Group in the United States and other countries.

Oracle is a registered trademark of Oracle Corporation.

PowerQuest and PartitionMagic are registered trademarks of PowerQuest Corporation in the United States and/or other countries.

RealNetworks, RealPlayer, and RealAudio are the registered trademarks of RealNetworks, Inc.

Red Hat, RPM, are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.

SAP, R/3, and mySAP are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Sun, Sun Microsystems, Java, Java Virtual Machine, JavaServer Pages, JDK, JRE, JSP, JVM, Netra, OpenJDK, Solaris, StarOffice, Sun Blade, Sun Enterprise, Sun Fire, SunOS, Ultra and VirtualBox are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Symantec and Ghost are registered trademarks of Symantec Corporation in the United States and other countries.

MATLAB is a registered trademark of The MathWorks, Inc.

SpeedTouch is a trademark of Thomson.

U.S. Robotics and Sportster are registered trademarks of U.S. Robotics Corporation.

VMware is a trademark of VMware, Inc.

Waterloo Maple and Maple are trademarks or registered trademarks of Waterloo Maple Inc.

Mathematica is a registered trademark of Wolfram Research, Inc.

XFree86 is a trademark of The XFree86 Project, Inc.

Ogg Vorbis and Xiph.Org are trademarks of Xiph.Org.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this document, and the FreeBSD Project was aware of the trademark claim, the designations have been followed by the „™” or the „®” symbol.

Spis treści

Przedmowa	xiii
I. Pierwsze kroki.....	xx
1. Wprowadzenie	1
1.1. Streszczenie	1
1.2. Witamy w świecie FreeBSD!	1
1.3. O Projekcie FreeBSD	4
2. Instalacja FreeBSD	9
2.1. Streszczenie	9
2.2. Czynności wstępne	9
2.3. Rozpoczęcie instalacji	15
2.4. Wprowadzenie do sysinstall	20
2.5. Przydział miejsca na dysku.....	25
2.6. Wybór składników instalacji.....	37
2.7. Wybór nośnika instalacji.....	39
2.8. Przystąpienie do instalacji	41
2.9. Po instalacji.....	42
2.10. Obsługiwany sprzęt.....	83
2.11. Rozwiązywanie problemów.....	83
2.12. Instalacja zaawansowana	86
2.13. Przygotowanie własnego nośnika instalacji.....	87
3. Podstawy Uniksa	93
3.1. Streszczenie	93
3.2. Konsole wirtualne i terminale	93
3.3. Prawa dostępu	96
3.4. Struktura katalogów.....	99
3.5. Organizacja dysku.....	101
3.6. Montowanie i odmontowywanie systemów plików	107
3.7. Procesy.....	110
3.8. Demony, sygnały i unicestwianie procesów	112
3.9. Powłoki	113
3.10. Edytory tekstu	115
3.11. Urządzenia i pliki urządzeń	116
3.12. Formaty binarne.....	116
3.13. Więcej informacji	118
4. Instalacja programów: pakiety i porty	121
4.1. Streszczenie	121
4.2. Omówienie instalacji oprogramowania	121
4.3. Odnalezienie programu dla siebie	123
4.4. Korzystanie z systemu pakietów.....	124
4.5. Korzystanie z kolekcji portów	126
4.6. Czynności po-instalacyjne	135
4.7. Jak radzić sobie ze źle przygotowanymi portami	135
5. System okien X	137
5.1. Streszczenie	137
5.2. Zrozumieć X	137

5.3. Instalacja X11	140
5.4. Konfiguracja X11	140
5.5. Korzystanie z czcionek w X11	144
5.6. Menedżer pulpitów X	148
5.7. Środowiska graficzne	151
II. Codzienne czynności	156
6. Aplikacje biurowe	157
6.1. Streszczenie	157
6.2. Przeglądarki internetowe	157
6.3. Programy codziennego użytku	161
6.4. Przeglądarki dokumentów	163
6.5. Finanse	165
6.6. Podsumowanie	166
7. Multimedia	168
7.1. Synopsis	168
7.2. Setting Up the Sound Card	168
7.3. MP3 Audio	172
7.4. Video Playback	174
7.5. Setting Up TV Cards	182
7.6. Image Scanners	184
8. Konfiguracja jądra FreeBSD	188
8.1. Streszczenie	188
8.2. Po co budować indywidualne jądro?	188
8.3. Budowanie i instalowanie indywidualnego jądra	189
8.4. Plik konfiguracyjny	191
8.5. Jeśli pojawią się kłopoty	204
9. Printing	206
9.1. Synopsis	206
9.2. Introduction	206
9.3. Basic Setup	207
9.4. Advanced Printer Setup	219
9.5. Using Printers	246
9.6. Alternatives to the Standard Spooler	253
9.7. Troubleshooting	253
10. Linux Binary Compatibility	257
10.1. Synopsis	257
10.2. Installation	257
10.3. Installing Mathematica®	260
10.4. Installing Maple™	262
10.5. Installing MATLAB®	264
10.6. Installing Oracle®	267
10.7. Installing SAP® R/3®	270
10.8. Advanced Topics	290

III. Administracja systemem.....	292
11. Configuration and Tuning.....	293
11.1. Synopsis.....	293
11.2. Initial Configuration.....	293
11.3. Core Configuration.....	294
11.4. Application Configuration.....	295
11.5. Starting Services.....	296
11.6. Configuring the cron Utility.....	298
11.7. Using rc under FreeBSD.....	300
11.8. Setting Up Network Interface Cards.....	301
11.9. Virtual Hosts.....	306
11.10. Configuration Files.....	307
11.11. Tuning with sysctl.....	311
11.12. Tuning Disks.....	312
11.13. Tuning Kernel Limits.....	315
11.14. Adding Swap Space.....	318
11.15. Power and Resource Management.....	319
11.16. Using and Debugging FreeBSD ACPI.....	320
12. The FreeBSD Booting Process.....	327
12.1. Synopsis.....	327
12.2. The Booting Problem.....	327
12.3. The Boot Manager and Boot Stages.....	328
12.4. Kernel Interaction During Boot.....	332
12.5. Device Hints.....	332
12.6. Init: Process Control Initialization.....	333
12.7. Shutdown Sequence.....	334
13. Użytkownicy i podstawy zarządzania kontami.....	335
13.1. Streszczenie.....	335
13.2. Wprowadzenie.....	335
13.3. Konto superużytkownika.....	337
13.4. Konta systemowe.....	337
13.5. Konta użytkowników.....	337
13.6. Modyfikacja ustawień kont.....	337
13.7. Ograniczanie użytkowników.....	341
13.8. Grupy.....	344
14. Security.....	346
14.1. Synopsis.....	346
14.2. Introduction.....	346
14.3. Securing FreeBSD.....	348
14.4. DES, MD5, and Crypt.....	354
14.5. One-time Passwords.....	355
14.6. TCP Wrappers.....	358
14.7. KerberosIV	361
14.8. Kerberos5	368
14.9. OpenSSL.....	375
14.10. VPN over IPsec.....	378
14.11. OpenSSH.....	389
14.12. File System Access Control Lists.....	395

14.13. Monitoring Third Party Security Issues	396
14.14. FreeBSD Security Advisories.....	397
14.15. Process Accounting	399
15. Mandatory Access Control.....	401
15.1. Synopsis.....	401
15.2. Key Terms in this Chapter	402
15.3. Explanation of MAC.....	403
15.4. Understanding MAC Labels	404
15.5. Planning the Security Configuration.....	409
15.6. Module Configuration.....	409
15.7. The MAC bsdextended Module.....	410
15.8. The MAC ifoff Module.....	411
15.9. The MAC portacl Module.....	412
15.10. The MAC partition Module	413
15.11. The MAC Multi-Level Security Module	414
15.12. The MAC Biba Module	416
15.13. The MAC LOMAC Module	417
15.14. Nagios in a MAC Jail.....	418
15.15. User Lock Down.....	421
15.16. Troubleshooting the MAC Framework.....	422
16. Security Event Auditing	424
16.1. Synopsis.....	424
16.2. Key Terms - Words to Know	424
16.3. Installing Audit Support	425
16.4. Audit Configuration.....	425
16.5. Event Audit Administration.....	428
17. Storage.....	430
17.1. Synopsis.....	430
17.2. Device Names	430
17.3. Adding Disks	431
17.4. RAID.....	433
17.5. USB Storage Devices.....	437
17.6. Creating and Using Optical Media (CDs)	439
17.7. Creating and Using Optical Media (DVDs)	444
17.8. Creating and Using Floppy Disks.....	449
17.9. Creating and Using Data Tapes	450
17.10. Backups to Floppies.....	452
17.11. Backup Strategies	453
17.12. Backup Basics.....	454
17.13. Network, Memory, and File-Backed File Systems	461
17.14. File System Snapshots	463
17.15. File System Quotas	464
17.16. Encrypting Disk Partitions.....	467
17.17. Encrypting Swap Space	473
18. GEOM: Modular Disk Transformation Framework.....	476
18.1. Synopsis.....	476
18.2. GEOM Introduction.....	476
18.3. RAID0 - Striping	476

18.4. RAID1 - Mirroring	478
18.5. GEOM Gate Network Devices	480
19. The Vinum Volume Manager	482
19.1. Synopsis	482
19.2. Disks Are Too Small	482
19.3. Access Bottlenecks	482
19.4. Data Integrity	484
19.5. Vinum Objects	485
19.6. Some Examples	487
19.7. Object Naming	493
19.8. Configuring Vinum	495
19.9. Using Vinum for the Root Filesystem	496
20. Localization - I18N/L10N Usage and Setup	503
20.1. Synopsis	503
20.2. The Basics	503
20.3. Using Localization	504
20.4. Compiling I18N Programs	509
20.5. Localizing FreeBSD to Specific Languages	510
21. The Cutting Edge	513
21.1. Synopsis	513
21.2. FreeBSD-CURRENT vs. FreeBSD-STABLE	513
21.3. Synchronizing Your Source	516
21.4. Rebuilding „world”	517
21.5. Tracking for Multiple Machines	529
IV. Komunikacja sieciowa	531
22. Serial Communications	532
22.1. Synopsis	532
22.2. Introduction	532
22.3. Terminals	536
22.4. Dial-in Service	541
22.5. Dial-out Service	549
22.6. Setting Up the Serial Console	552
23. PPP and SLIP	560
23.1. Synopsis	560
23.2. Using User PPP	560
23.3. Using Kernel PPP	572
23.4. Troubleshooting PPP Connections	579
23.5. Using PPP over Ethernet (PPPoE)	582
23.6. Using PPP over ATM (PPPoA)	584
23.7. Using SLIP	587
24. Electronic Mail	596
24.1. Synopsis	596
24.2. Using Electronic Mail	596
24.3. sendmail Configuration	598
24.4. Changing Your Mail Transfer Agent	601
24.5. Troubleshooting	604
24.6. Advanced Topics	606

24.7. SMTP with UUCP	608
24.8. Setting Up to Send Only	610
24.9. Using Mail with a Dialup Connection	611
24.10. SMTP Authentication	612
24.11. Mail User Agents	613
24.12. Using fetchmail	620
24.13. Using procmail	621
25. Network Servers	623
25.1. Synopsis	623
25.2. The inetd „Super-Server”	623
25.3. Network File System (NFS)	627
25.4. Network Information System (NIS/YP)	632
25.5. Automatic Network Configuration (DHCP)	647
25.6. Domain Name System (DNS)	652
25.7. Apache HTTP Server	661
25.8. File Transfer Protocol (FTP)	664
25.9. File and Print Services for Microsoft® Windows clients (Samba)	665
25.10. Clock Synchronization with NTP	668
26. Firewalls	671
26.1. Introduction	671
26.2. Firewall Concepts	671
26.3. Firewall Packages	672
26.4. The OpenBSD Packet Filter (PF) and ALTQ	672
26.5. The IPFILTER (IPF) Firewall	674
26.6. IPFW	693
27. Advanced Networking	711
27.1. Synopsis	711
27.2. Gateways and Routes	711
27.3. Wireless Networking	717
27.4. Bluetooth	730
27.5. Bridging	738
27.6. Diskless Operation	741
27.7. ISDN	747
27.8. Network Address Translation	751
27.9. Parallel Line IP (PLIP)	754
27.10. IPv6	756
27.11. Asynchronous Transfer Mode (ATM)	761
V. Dodatki	763
A. Obtaining FreeBSD	764
A.1. CDROM and DVD Publishers	764
A.2. FTP Sites	766
A.3. Anonymous CVS	776
A.4. Using CTM	779
A.5. Using CVSup	783
A.6. Using Portsnap	804
A.7. CVS Tags	806
A.8. AFS Sites	811

A.9. rsync Sites	811
B. Bibliografia	813
B.1. Książki i czasopisma poświęcone FreeBSD	813
B.2. Podręczniki użytkowania	814
B.3. Podręczniki administracji.....	814
B.4. Podręczniki programowania.....	815
B.5. Komponenty systemu operacyjnego	816
B.6. Bezpieczeństwo.....	816
B.7. Sprzęt	816
B.8. Historia systemów UNIX.....	817
B.9. Czasopisma	817
C. Resources on the Internet	818
C.1. Mailing Lists	818
C.2. Usenet Newsgroups.....	834
C.3. World Wide Web Servers	835
C.4. Email Addresses.....	841
C.5. Shell Accounts	841
D. Klucze PGP	843
D.1. Oficerowie.....	843
D.2. Członkowie głównego zespołu projektantów	843
D.3. Twórcy	845
FreeBSD Glossary	918
Colophon.....	942

Spis tabel

2-1. Przykładowa lista urządzeń	10
2-2. Układ partycji pierwszego dysku	31
2-3. Układ partycji dla kolejnych dysków	32
2-4. Dostępne profile zabezpieczeń	50
2-5. Nazwy obrazów ISO dla FreeBSD 4.x i ich znaczenie.....	88
2-6. Nazwy obrazów ISO dla FreeBSD 5.x i ich znaczenie.....	88
3-1. Oznaczenia dysków	106
17-1. Physical Disk Naming Conventions	430
19-1. Vinum Plex Organizations	486
22-1. DB-25 to DB-25 Null-Modem Cable	533
22-2. DB-9 to DB-9 Null-Modem Cable	533
22-3. DB-9 to DB-25 Null-Modem Cable	534
22-4. Signal Names	542
27-1. Wiring a Parallel Cable for Networking	754
27-2. Reserved IPv6 addresses	757

Przedmowa

Docelowy czytelnik

Osoba poznająca dopiero system FreeBSD odnajdzie w pierwszej części niniejszej książki szereg porad prowadzących użytkownika przez proces instalacji i delikatnie prezentujących pewne koncepcje i konwencje stojące u podstaw systemów UNIX®. Przebrnięcie przez tę część wymaga niewiele więcej niż chęć poznania i umiejętność przyswajania sobie nowych koncepcji w miarę jak będą one prezentowane.

Po dotrzeniu do drugiej, zdecydowanie obszerniejszej części Podręcznika, czytelnik będzie miał do dyspozycji pełną wiedzę z zakresu wszystkich zagadnień znajdujących się w polu zainteresowań administratorów systemów FreeBSD. Niektóre z zawartych tutaj rozdziałów mogą wymagać wcześniejszego zapoznania się z odpowiednią literaturą. W takich przypadkach, będzie to wyszczególnione w streszczeniu na początku każdego rozdziału.

Dodatek B zawiera listę dodatkowych źródeł informacji.

Zmiany od wydania drugiego

Niniejsze trzecie wydanie stanowi punkt kulminacyjny przeszło dwuletniej pracy oddanych członków Projektu Dokumentacji FreeBSD. Główne zmiany jakie w tym okresie zostały dokonane to:

- Rozdział 11, Konfiguracja i dostrajanie został poszerzony o nowe informacje o zarządzaniu mocą i zasobami APCI, opis narzędzia cron i kolejną porcję opcji dostrajania jądra.
- Rozdział 14, Bezpieczeństwo, został poszerzony o nowe informacje odnośnie wirtualnych sieci prywatnych (VPN), list kontroli dostępu do systemu plików, i buletynach bezpieczeństwa.
- Rozdział 15, Mandatory Access Control (MAC), is a new chapter with this edition. It explains what MAC is and how this mechanism can be used to secure a FreeBSD system.
- Rozdział 17, Storage, has been expanded with new information about USB storage devices, file system snapshots, file system quotas, file and network backed filesystems, and encrypted disk partitions.
- Rozdział 19, Vinum, is a new chapter with this edition. It describes how to use Vinum, a logical volume manager which provides device-independent logical disks, and software RAID-0, RAID-1 and RAID-5.
- A troubleshooting section has been added to Rozdział 23, PPP and SLIP.
- Rozdział 24, Electronic Mail, has been expanded with new information about using alternative transport agents, SMTP authentication, UUCP, fetchmail, procmail, and other advanced topics.
- Rozdział 25, Network Servers, is all new with this edition. This chapter includes information about setting up the Apache HTTP Server, FTPd, and setting up a server for Microsoft Windows clients with Samba. Some sections from Rozdział 27, Advanced Networking, were moved here to improve the presentation.
- Rozdział 27, Advanced Networking, has been expanded with new information about using Bluetooth devices with FreeBSD, setting up wireless networks, and Asynchronous Transfer Mode (ATM) networking.
- Definicje i wykorzystywane w książce terminy techniczne zostały zebrane razem w formie leksykonu.
- Dokonano wielu estetycznych poprawek tabel i rysunków.

Zmiany od wydania pierwszego

Wydanie drugie stanowiło punkt kulminacyjny przeszło dwuletniej pracy oddanych członków Projektu Dokumentacji FreeBSD. Główne zmiany jakie w tym okresie zostały dokonane to:

- Dodano indeks.
- Wszystkie diagramy ASCII zostały zastąpione rysunkami graficznymi.
- Dodano standardowe streszczenie do wszystkich rozdziałów, informujące jakie informacje rozdział zawiera i co powinien wiedzieć czytelnik nim przystąpi do czytania.
- Zawartość podręcznika została zreorganizowana w trzy logiczne części: „Pierwsze kroki”, „Administracja systemem” oraz „Dodatki”.
- Rozdział 2 („Instalacja FreeBSD”) został całkowicie przepisany na nowo. Dołączono wiele zrzutów ekranu, by ułatwić nowym użytkownikom przyswojenie tekstu.
- Rozdział 3 („Podstawy Uniksa”) został poszerzony o dodatkow informacje o procesach, demonach i sygnałach.
- Rozdział 4 („Instalacja programów”) został poszerzony o dodatkowe informacje o zarządzaniu pakietami binarnymi.
- Rozdział 5 („System okien X”) został w całości napisany od nowa kładąc nacisk na współczesne środowiska graficzne we XFree86™ 4.X, takie jak **KDE** i **GNOME**.
- Rozdział 12 („Proces uruchamiania FreeBSD”) został poszerzony.
- Rozdział 17 („Pamięć”) został napisany na podstawie rozdziałów „Dyski” oraz „Kopie zapasowe”. Uważamy, że zagadnienia te łatwiej jest zrozumieć, gdy są przedstawiane jako jeden rozdział. Dodano również podrozdział traktujący o RAID (zarówno sprzętowym jak i programowym).
- Rozdział 22 („Komunikacja szeregową”) został całkowicie zreorganizowany i zaktualizowany dla FreeBSD 4.X/5.X.
- Rozdział 23 („PPP i SLIP”) zostały zasadniczo zaktualizowane.
- Rozdział 27 („Advanced Networking”) został zaktualizowany.
- Rozdział 24 („Poczta elektroniczna”) został rozszerzony materiały traktujące o konfiguracji programu **sendmail**.
- Rozdział 10 („Kompatybilność z Linuksem”) został poszerzony o informacje o instalacji bazy **Oracle®** oraz **SAP® R/3®**.
- W drugim wydaniu dodano nowe rozdziały:
 - Konfiguracja i dostrajanie (Rozdział 11).
 - Multimedia (Rozdział 7)

Układ książki

Niniejsza książka została podzielona na pięć logicznych części. Część pierwsza, *Pierwsze kroki*, opisuje proces instalacji oraz podstawy użytkowania systemu FreeBSD. Zaleca się aby czytelnik zapoznał się z tymi rozdziałami kolejno, pomijając jedynie znane tematy. Część druga, *Codzienne czynności*, prezentuje niektóre z najczęściej wykorzystywanych funkcji FreeBSD. Ta część, wraz kolejnymi, może być czytania bez określonej kolejności. Każdy

z wchodzących w jej skład rozdziałów zaczyna się od zwięzłego streszczenia zawartości i przedstawienia co czytelnik powinien już wiedzieć. Celem takiego układu jest pozwolenie zwykłemu czytelnikowi pominąć pewne rozdziały, by przejść od razu do najbardziej interesujących. Część trzecia, *Administracja Systemem*, opisuje zagadnienia administracyjne. Część czwarta, *Komunikacja sieciowa*, zawiera tematy związane z pracą w sieci oraz obsługą serwerów. Część piąta zawiera dodatki.

Rozdział 1, Wprowadzenie

Wprowadza nowego użytkownika w świat FreeBSD. Streszcza historię Projektu FreeBSD, stawiane przed nim cele oraz model rozwoju.

Rozdział 2, Instalacja

Przeprowadza użytkownika przez cały proces instalacji. Opisuje również kilka zaawansowanych zagadnień, jak np. instalację przez konsolę szeregową.

Rozdział 3, Podstawy Uniksa

Przedstawia podstawowe polecenie i funkcje systemu operacyjnego FreeBSD. Jeśli pracowaliśmy w Linuksie bądź w innym systemie typu UNIX najprawdopodobniej możemy pominąć ten rozdział.

Rozdział 4, Instalacja programów

Opisuje metody instalacji dodatkowego oprogramowania we FreeBSD za pomocą systemu „Kolekcji portów” oraz typowych pakietów binarnych.

Rozdział 5, System okien X

Opisuje ogólnie System okien X oraz wykorzystanie X11 we FreeBSD. Ponadto, przedstawia typowe środowiska graficzne jak np. **KDE** czy **GNOME**.

Rozdział 6, Aplikacje biurowe

Lists some common desktop applications, such as web browsers and productivity suites, and describes how to install them on FreeBSD.

Rozdział 7, Multimedia

Shows how to set up sound and video playback support for your system. Also describes some sample audio and video applications.

Rozdział 8, Configuring the FreeBSD Kernel

Explains why you might need to configure a new kernel and provides detailed instructions for configuring, building, and installing a custom kernel.

Rozdział 9, Printing

Describes managing printers on FreeBSD, including information about banner pages, printer accounting, and initial setup.

Rozdział 10, Linux® Binary Compatibility

Describes the Linux compatibility features of FreeBSD. Also provides detailed installation instructions for many popular Linux applications such as **Oracle**, **SAP R/3**, and **Mathematica®**.

Rozdział 11, Configuration and Tuning

Describes the parameters available for system administrators to tune a FreeBSD system for optimum performance. Also describes the various configuration files used in FreeBSD and where to find them.

Rozdział 12, Booting Process

Describes the FreeBSD boot process and explains how to control this process with configuration options.

Rozdział 13, Users and Basic Account Management

Describes the creation and manipulation of user accounts. Also discusses resource limitations that can be set on users and other account management tasks.

Rozdział 14, Security

Describes many different tools available to help keep your FreeBSD system secure, including Kerberos, IPsec and OpenSSH.

Rozdział 15, Mandatory Access Control

Explains what Mandatory Access Control (MAC) is and how this mechanism can be used to secure a FreeBSD system.

Rozdział 17, Storage

Describes how to manage storage media and filesystems with FreeBSD. This includes physical disks, RAID arrays, optical and tape media, memory-backed disks, and network filesystems.

Rozdział 18, GEOM

Describes what the GEOM framework in FreeBSD is and how to configure various supported RAID levels.

Rozdział 19, Vinum

Describes how to use Vinum, a logical volume manager which provides device-independent logical disks, and software RAID-0, RAID-1 and RAID-5.

Rozdział 20, Localization

Describes how to use FreeBSD in languages other than English. Covers both system and application level localization.

Rozdział 21, The Cutting Edge

Explains the differences between FreeBSD-STABLE, FreeBSD-CURRENT, and FreeBSD releases. Describes which users would benefit from tracking a development system and outlines that process.

Rozdział 22, Serial Communications

Explains how to connect terminals and modems to your FreeBSD system for both dial in and dial out connections.

Rozdział 23, PPP and SLIP

Describes how to use PPP, SLIP, or PPP over Ethernet to connect to remote systems with FreeBSD.

Rozdział 24, Electronic Mail

Explains the different components of an email server and dives into simple configuration topics for the most popular mail server software: **sendmail**.

Rozdział 25, Network Servers

Provides detailed instructions and example configuration files to set up your FreeBSD machine as a network filesystem server, domain name server, network information system server, or time synchronization server.

Rozdział 26, Firewalls

Explains the philosophy behind software-based firewalls and provides detailed information about the configuration of the different firewalls available for FreeBSD.

Rozdział 27, Advanced Networking

Describes many networking topics, including sharing an Internet connection with other computers on your LAN, advanced routing topics, wireless networking, bluetooth, ATM, IPv6, and much more.

Dodatek A, Obtaining FreeBSD

Lists different sources for obtaining FreeBSD media on CDROM or DVD as well as different sites on the Internet that allow you to download and install FreeBSD.

Dodatek B, Bibliography

This book touches on many different subjects that may leave you hungry for a more detailed explanation. The bibliography lists many excellent books that are referenced in the text.

Dodatek C, Resources on the Internet

Describes the many forums available for FreeBSD users to post questions and engage in technical conversations about FreeBSD.

Dodatek D, PGP Keys

Lists the PGP fingerprints of several FreeBSD Developers.

Konwencje użyte w tej książce

W celu utrzymania jednolitości i łatwości czytania niniejszego tekstu w książce zastosowane zostały następujące konwencje.

Konwencje typograficzne

Kursywa

Czcionka *pochyła* stosowana jest do wskazania plików, adresów URL, szczególnie akcentowanych fragmentów i pierwszego zastosowania zwrotów technicznych.

Stała szerokość

Czcionka o stałej szerokości stosowana jest do przedstawienia komunikatów o błędach, poleceń, zmiennych środowiskowych, nazw portów, nazw komputerów, nazw użytkowników i grup, nazw urządzeń, zmiennych i fragmentów kodu.

Pogrubienie

Czcionka **pogrubiona** stosowana jest do nazw programów, poleceń i klawiszy.

Zadania użytkownika

Zgodnie z konwencją typograficzną, klawisze, które ma nacisnąć użytkownik w trakcie pracy z opisywanym programem, zostały oznaczone **pogrubieniem** by wyróżniały się z reszty tekstu. Kombinacje klawiszy, które należy nacisnąć jednocześnie zawierają znak ‘+’ pomiędzy, np.:

Ctrl+Alt+Del

Oznacza, że użytkownik powinien nacisnąć **Ctrl**, **Alt** i **Del** jednocześnie.

Klawisze, które należy nacisnąć kolejno będą oddzielone przecinkiem, np.:

Ctrl+X, Ctrl+S

Co oznacza, że użytkownik powinien nacisnąć klawisze **Ctrl** i **X** jednocześnie, a następnie **Ctrl** i **S**.

Przykłady

Przykłady zaczynające się od `E:\>` wskazują polecenie systemu MS-DOS®. Jeśli nie jest wyraźnie zaznaczone, że jest inaczej, polecenia te mogą być wprowadzane bezpośrednio w oknie „Linii poleceń” w środowisku Microsoft® Windows®.

```
E:\> tools\fdimage floppies\kern.flp A:
```

Przykłady zaczynające się od `#` wskazują polecenie, które musi być wprowadzone przez użytkownika z uprawnieniami administratora systemu FreeBSD. Możesz zalogować się jako `root` i wprowadzić polecenie, bądź zalogować jako zwykły użytkownik i wykorzystać `su(1)` by uzyskać prawa administratora.

```
# dd if=kern.flp of=/dev/fd0
```

Przykłady zaczynające się od `%` wskazują, iż polecenie powinno być wprowadzone przez zwykłego użytkownika. Jeśli nie jest inaczej zaznaczone, stosowana jest składnia powłoki C (`ssh`) do ustawiania zmiennych środowiskowych i uruchamiania innych poleceń powłoki.

```
% top
```

Podziękowania

Niniejsza książka jest efektem pracy setek ludzi z całego świata. Niezależnie czy przysłali poprawkę literówki czy cały rozdział, każdy wkład jest doceniany.

Kilka firm wsparło rozwój tego dokumentu opłacając autorów, by mogli pracować nad nią w pełnym wymiarze czasowym, finansując publikację w formie papierowej, itd. Pragniemy wymienić przede wszystkim BSDi (przejęte później przez Wind River Systems (<http://www.windriver.com>)), które opłaciło pracę członków Projektu

Dokumentacji FreeBSD nad korektami książki, przygotowując ją do pierwszej publikacji drukowanej w Marcu 2000 r. (ISBN 1-57176-241-8). Następnie, Wind River Systems sfinansowało pracę kolejnych osób przygotowujących nowe rozdziały, a także format wydruku. Kulminacją ich pracy jest drugie wydanie, które ujrzało światło dzienne w Listopadzie 2001 r. (ISBN 1-57176-303-1). W latach 2003-2004, FreeBSD Mall, Inc (<http://www.freebsdmall.com>) sfinansowało prace nad korektą Podręcznika, przygotowywanego do trzeciego wydania w postaci drukowanej.

I. Pierwsze kroki

Ta część Podręcznika FreeBSD adresowana jest do użytkowników i administratorów, którzy nie mieli dotychczas kontaktu z systemem FreeBSD. Niniejsze rozdziały mają za zadanie:

- Zaprezentować system FreeBSD.
- Przeprowadzić przez proces instalacji.
- Nauczyć podstaw systemu UNIX.
- Pokazać jak zainstalować programy innych autorów, dostępne w ogromnej ilości dla systemu FreeBSD.
- Przedstawić system X - system okien UNIX, oraz szczegółowo wyjaśnić jak prawidłowo skonfigurować środowisko graficzne, tak by zwiększyć efektywność swej pracy.

Staraliśmy się sprowadzić liczbę odnośników wewnątrz tekstu do możliwie najmniejszej, tak by zminimalizować ilość „przeskoków” i ułatwić czytanie Podręcznika od deski do deski.

Rozdział 1. Wprowadzenie

Rozdział zreorganizował i częściowo napisał od nowa Jim Mock. Tłumaczył Cezary Morga.

1.1. Streszczenie

Dziękujemy za zainteresowanie FreeBSD! W niniejszym rozdziale opisane zostaną różne aspekty Projektu FreeBSD, takie jak jego historia, obrany cel, czy model rozwoju.

Czytając ten rozdział poznamy:

- Zależności istniejące między FreeBSD i innymi systemami operacyjnymi.
- Historię Projektu FreeBSD.
- Cele stawiane przed Projektem FreeBSD.
- Podstawowe zagadnienia związane z modelem rozwoju otwartego oprogramowania (ang. open source) FreeBSD.
- I oczywiście, dowiemy się skąd pochodzi nazwa „FreeBSD”.

1.2. Witamy w świecie FreeBSD!

FreeBSD jest systemem operacyjnym bazującym na 4.4BSD-Lite, a przeznaczonym dla komputerów pracujących na platformach Intelu (x86 i Itanium®), AMD64, Alpha™ oraz Sun UltraSPARC®. Przygotowywane są również wersje dla innych platform. Więcej informacji dostępnych jest w historii FreeBSD bądź w nocie o aktualnym wydaniu. Jeśli chciałbyś wspomóc rozwój Projektu (np. kod źródłowy, sprzęt, nieoznakowane banknoty) przeczytaj artykuł o współpracy z Projektem FreeBSD (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/contributing/index.html) (ang.).

1.2.1. Co potrafi FreeBSD?

Tłumaczył Aleksander Fafuła. Przekład uzupełnił Cezary Morga.

FreeBSD posiada mnóstwo zalet. Oto niektóre z nich:

- *Wielozadaniowość z wyłuszczeniem*, wraz z dynamiczną regulacją priorytetów, by zapewnić sprawne i bezkonfliktowe współdzielenie zasobów komputera przez aplikacje oraz użytkowników, nawet w sytuacjach największego obciążenia systemu.
- *Wieloużytkownikowość* pozwalająca na jednoczesne wykorzystanie komputera z systemem FreeBSD przez wielu użytkowników. Oznacza to, np. prawidłowe dzielenie dostępu do urządzeń zewnętrznych jak np. do drukarki, pomiędzy wszystkich użytkowników lokalnych jak i sieciowych. Ograniczenia dostępu do zasobów mogą być definiowane dla konkretnych użytkowników bądź grup użytkowników, co z kolei pozwala na zabezpieczenie krytycznych zasobów systemowych przed nadużyciami.
- Pełna obsługa *sieci TCP/IP*, oraz innych sieciowych standardów jak SLIP, PPP, NFS, DHCP czy NIS. Oznacza to, że twój system FreeBSD może bez problemów współpracować z dowolnymi innymi systemami operacyjnymi, jak również pracować w roli serwera w przedsiębiorstwie, dostarczając niezbędnych funkcji jak np. NFS (zdalny dostęp do plików) wraz z obsługą emaila, bądź pozwoli na umieszczenie internetowej wizytówki twojej

organizacji na stronie WWW czy dokumentów na serwerze FTP. Może również realizować przekierowywanie (ruting) pakietów, a także pełnić rolę zapory ogniowej (firewall).

- *Ochrona pamięci* gwarantuje, że programy (bądź użytkownicy) nie mogą ingerować w pracę innych aplikacji. Innymi słowy, awaria danego programu w żaden sposób nie wpływa na działanie pozostałych.
- FreeBSD jest 32-bitowym systemem operacyjnym (64-bitowym na platformach Alpha, Itanium, AMD64 i UltraSPARC) i właśnie jako taki projektowany był od początku.
- Obecnie standardowy *System okien X* (X11R6; X Window System) dostarcza interfejsu graficznego (GUI) w cenie zwykłej karty VGA i monitora. Ponadto dostępny jest z pełnym kodem źródłowym.
- *Zgodność binarna* z wieloma systemami typu UNIX. FreeBSD posiada możliwość uruchamiania programów skompilowanych dla Linuksa, SCO, SVR4, BSDI i NetBSD.
- Tysiące aplikacji *gotowych do pracy*, dostępnych z kolekcji *portów* i *pakietów* FreeBSD. Czemu szukać w sieci, skoro wszystko można znaleźć właśnie tutaj?
- Tysiące dodatkowych i *łatwych do przeniesienia* programów dostępnych w Internecie. FreeBSD jest zgodny z wieloma popularnymi, nawet komercyjnymi systemami typu UNIX i tym samym większość programów wymaga zaledwie kilku, jeśli w ogóle, zmian w kodzie aby poprawnie skompilować i uruchomić.
- Stroniciwana *pamięć wirtualna* oraz współdzielona pamięć podręczna „VM/buffer cache” zaprojektowane by efektywnie zaspokajać potrzeby aplikacji z dużym apetytem na pamięć, przy jednoczesnym zapewnieniu ciągłej interakcji systemu z użytkownikami.
- Wsparcie dla technologii *SMP*, dla maszyn z wieloma procesorami.
- Kompletne środowiska programistyczne dla języków *C*, *C++* i *Fortran*. FreeBSD posiada również wiele dodatkowych środowisk dla innych języków programowania dostępnych w kolekcji portów i pakietów.
- Dostępność *kodu źródłowego* dla całego systemu oznacza, iż to właśnie ty posiadasz największą kontrolę nad swoim środowiskiem pracy. Czemu zamykać się w kręgu rozwiązań własnościowych i być skazanym na łaskę dostawcy systemu, kiedy można mieć prawdziwie otwarty system?
- Obszerłą *dokumentację* dostępną w Internecie..
- *I wiele więcej!*

FreeBSD jest oparty na systemie 4.4BSD-Lite pochodzącym z Computer Systems Research Group (CSRG) z Uniwersytetu Kalifornijskiego w Berkeley. Podtrzymuje dostojną tradycję trendu rozwojowego systemów BSD. Oprócz doskonałej pracy wykonanej przez CSRG również programiści z Projektu FreeBSD spędzili dodatkowe tysiące godzin, aby udoskonalić go i przygotować na trudne, życiowe sytuacje. W czasie gdy wielu z komercyjnych gigantów branży komputerów PC stara się wyposażać swoje systemy operacyjne w podobne cechy, by osiągnąć takie same wyniki i poziom niezawodności, FreeBSD oferuje to już *teraz!*

Liczba aplikacji z którymi może współpracować FreeBSD jest ograniczona jedynie przez naszą wyobraźnię. Od projektów programistycznych, poprzez automatyzację produkcji w fabrykach, kontrolę stanu magazynów, po regulację azymutu anteny satelitarnej; jeśli jest to możliwe w komercyjnych systemach UNIX jest to więcej niż prawdopodobne, że możesz to zrobić również we FreeBSD! On sam korzysta z dosłownie tysięcy doskonale dopracowanych aplikacji, nierzadko pochodzących z komercyjnych centrów projektowych bądź laboratoriów uniwersyteckich, dostępnych niemalże bądź całkowicie za darmo. Dostępne jest również oprogramowanie komercyjne, którego liczba rośnie równie szybko, jak oprogramowania bezpłatnego.

Jako, że kod źródłowy FreeBSD jest publicznie dostępny, system może zostać dostosowany do wielu specjalistycznych projektów oraz zastosowań, co jest niemożliwe w przypadku wielu systemów komercyjnych. Oto krótka lista aplikacji, z którymi najczęściej używany jest FreeBSD:

- *Usługi internetowe*: doskonała obsługa TCP/IP wbudowana we FreeBSD, czyni go idealną platformą dla szeregu usług internetowych, na przykład:
 - Serwery FTP
 - Serwery witryn WWW (standardowe bądź zabezpieczone [SSL])
 - Zapory ogniowe i bramy NAT („maskarada IP”)
 - Serwery poczty elektronicznej
 - Serwery USENET bądź systemy Forum
 - I więcej...

Wraz z FreeBSD możesz zacząć świadczyć usługi internetowe już na niedrogim komputerze PC klasy 386 i rozwijać bazę sprzętową swojego przedsiębiorstwa aż do cztero-procesorowego Xeona z macierzą RAID.

- *Edukacja*: jesteś studentem informatyki bądź pokrewnej dziedziny techniki? Nie ma lepszego sposobu na poznanie systemu operacyjnego, architektury komputerów oraz zagadnień sieciowych niż poprzez doświadczenie, które daje praca z FreeBSD. Duża liczba darmowych programów typu CAD, matematycznych czy graficznych będzie wysoce użyteczna dla tych, których głównym zainteresowaniem w komputerach jest aby zmusić je do pracy *za nas!*
- *Badania*: oferując dostęp do kodu źródłowego całego systemu, FreeBSD stanowi doskonałą platformę dla prowadzenia badań nad systemami operacyjnymi oraz innymi dziedzinami nauk komputerowych. Idea otwartego źródła wspomaga także całe grupy współpracujące zdalnie nad różnymi zadaniami, pomagając zapomnieć im o problemach związanych ze specjalnymi warunkami licencyjnymi oraz ograniczeniami.
- *Sieć*: potrzebujesz nowego rutera? Serwera nazw (DNS)? Zapory ogniowej (firewalla), by wystrzec się niepowołanych użytkowników w swojej sieci wewnętrznej? FreeBSD może w łatwy sposób zamienić bezużytecznego 486 lub nawet 386, stojącego w kącie, w zaawansowany router z wyszukanyymi opcjami filtrowania pakietów.
- *Środowisko graficzne*: FreeBSD stanowi dobre rozwiązanie dla niedrogiego terminala graficznego. W tym celu można wykorzystać dostępny serwer X11, bądź jeden z doskonałych komercyjnych serwerów Xi Graphics (<http://www.xig.com>). W przeciwieństwie do typowych terminali graficznych, FreeBSD pozwala na uruchamianie wielu aplikacji lokalnie jeśli zajdzie taka potrzeba, odciążając tym samym główny serwer. FreeBSD może być również uruchamiany w systemach „bezdyskowych” zmniejszając tym samym cenę komputerów służących za terminale.
- *Programowanie*: system FreeBSD zaopatrzony jest w pełen zestaw narzędzi programistycznych, włączając w to sławny kompilator oraz debugger GNU C/C++.

FreeBSD jest dostępny zarówno w postaci kodu źródłowego jak i skompilowanych binariów dostępnych na płytach CDROM, DVD i poprzez anonimowy serwer FTP. Dodatek A zawiera więcej informacji nt. sposobów uzyskania FreeBSD.

1.2.2. Kto używa FreeBSD?

FreeBSD zasila niektóre z największych witryn w Internecie, m.in:

- Yahoo! (<http://www.yahoo.com/>)
- Apache (<http://www.apache.org/>)

- Blue Mountain Arts (<http://www.bluemountain.com/>)
- Pair Networks (<http://www.pair.com/>)
- Sony Japan (<http://www.sony.co.jp/>)
- Netcraft (<http://www.netcraft.com/>)
- Weathernews (<http://www.wni.com/>)
- Supervalu (<http://www.supervalu.com/>)
- TELEHOUSE America (<http://www.telehouse.com/>)
- Sophos Anti-Virus (<http://www.sophos.com/>)
- JMA Wired (<http://www.jmawired.com/>)

i wiele więcej.

1.3. O Projekcie FreeBSD

Niniejszy podrozdział zawiera podstawowe informacje o projekcie, m.in. krótką historię, cele stawiane przed projektem i stosowany model rozwoju.

1.3.1. Krótka historia FreeBSD

Napisał Jordan Hubbard.

Genezy projektu FreeBSD należy doszukiwać się w pierwszej połowie roku 1993. Wyrósł on częściowo z „Nieoficjalnego zestawu łat dla 386BSD” (patchkit). Stworzony został przez trzech ostatnich koordynatorów zestawu: Nate’a Williamsa, Roda Grimesa i mnie.

Naszym pierwotnym celem było przygotowanie migawki z rozwoju 386BSD, wprowadzającej szereg poprawek, których mechanizm zestawu łat nie był w stanie zrealizować. Niektórzy z czytających mogą pamiętać wczesną nazwę projektu „386BSD 0.5” bądź „386BSD Interim”.

386BSD był systemem operacyjnym Billa Jolitza, cierpiącym w tym okresie z powodu przeszło rocznego zastoju. W wyniku puchnięcia zestawu łat z dnia na dzień coraz bardziej, jednomyślnie postanowiliśmy spróbować naprawić sytuację. Zdecydowaliśmy się wspomóc Billa dostarczając owej „porządkującej” migawki. Niestety plan spalił na panewce gdy Bill Jolitz nagle zdecydował cofnąć swoje poparcie dla projektu, nie informując co zamierza wprowadzić w jego miejsce.

Szybko stwierdziliśmy, że rozpoczęte zadanie jest warte świeczki nawet bez wsparcia Billa. Tym samym przyjęliśmy nazwę „FreeBSD” ukutą przez Davida Greenmana. Cele projektu zostały wstępnie określone po rozmowach z ówczesnymi użytkownikami systemu. Gdy stało się jasne, że projekt zmierza w kierunku stania się rzeczywistością, skontaktowałem się z firmą Walnut Creek CDROM w celu usprawnienia metod dystrybucji FreeBSD, szczególnie z myślą o tych nieszczęśnikach, którzy mieli utrudniony dostęp do Internetu. Walnut Creek CDROM nie tylko wsparł pomysł dystrybucji FreeBSD na płytach CD, ale również wyszedł nam na przeciw oferując projektowi maszynę do pracy i szybkie łącze z Internetem. Jest mało prawdopodobne, że projekt zaszedł by aż tak daleko bez niespotykanej wręcz wiary Walnut Creek CDROM w kompletnie mało znany projekt, którym w owym czasie był FreeBSD.

Pierwszą wersją rozprowadzaną na płytach CD (a także w Internecie) był FreeBSD 1.0, wydany w grudniu 1993 r. Oparty był on bezpośrednio na 4.3BSD-Lite („Net/2”) z Uniwersytetu Kalifornijskiego w Berkeley. Zawierał

również wiele dodatkowych aplikacji pochodzących z 386BSD oraz Free Software Foundation. Można przyjąć, iż osiągnął on całkiem rozsądny sukces jak na pierwszą wersję. Następujące po nim wydanie FreeBSD 1.1 w maju 1994 r. było pełnym sukcesem.

Mniej więcej w tym właśnie czasie czarne chmury niespodzianie pojawiły się nad horyzontem. Powodem tego była ugoda w przeciągającym się procesie pomiędzy Novellem i Uniwersytetem w Berkeley odnośnie legalności kalifornijskiego Net/2. Jednym z warunków ugody było ustępstwo Berkeley stwierdzające, iż znaczne części kodu Net/2 zostały „powielone” z kodu systemu UNIX, będącego własnością Novella, który z kolei nabył go wcześniej od AT&T. W zamian Berkeley uzyskało „błogosławieństwo” Novella w pracach nad 4.4BSD-Lite i zapewnienie, że gdy się w końcu pojawi nie będzie określone jako kopia kodu Novella. Ponadto wszyscy użytkownicy Net/2 mieli być gorąco zachęceni do aktualizacji systemu. Ugoda ta dotyczyła również FreeBSD, bowiem projekt miał wstrzymać dystrybucję swoich produktów bazujących na Net/2 do końca lipca 1994 r. Zgodnie z warunkami porozumienia, pozwolono projektowi na jedno ostatnie wydanie przed tym terminem. Było to FreeBSD 1.1.5.1.

Rozpoczęła się żmudna praca nad ponownym stworzeniem FreeBSD z części całkowicie nowego i raczej niekompletnego 4.4BSD-Lite. Wydanie „Lite” było w rzeczy samej „lekkie”; częściowo w wyniku usunięcia przez CSRG Uniwersytetu w Berkeley wielkich partii kodu (z uwagi na pewne wymogi prawne), które odpowiadały za przygotowanie samodzielnie uruchamiającego się systemu, oraz z faktu, że wersja 4.4 nie była jeszcze gotowa na platformę Intel. Prace potrwały do listopada 1994 r., kiedy to wydany został FreeBSD 2.0, rozprowadzany zarówno przez sieć jak i na płytach CD (w późnym grudniu). Pomimo kilku niedociągnięć wydanie osiągnęło znaczący sukces. Przy czym już w styczniu 1995 r. zostało zastąpione stabilniejszym i łatwiejszym w instalacji FreeBSD 2.0.5.

FreeBSD 2.1.5 wydaliśmy w sierpniu 1996. Wersja ta zyskała popularność szczególnie pośród dostawców usług internetowych (ISP) oraz szerokopojętej społeczności komercyjnej. Docenione zostało również kolejne wydanie w gałęzi 2.1-STABLE. Mowa tu o FreeBSD 2.1.7.1, wydanym w lutym 1997 r., a zamykającym główne prace nad 2.1-STABLE. Od tej pory trwały jedynie prace nad utrzymaniem gałęzi (RELENG_2_1_0); dodawane były łatwy bezpieczeństwa i naprawiane krytyczne luki.

Z głównego nurtu rozwoju („-CURRENT”) w listopadzie 1996 r. odgałęził się FreeBSD 2.2 jako gałąź RELENG_2_2. Pierwsze pełne wydanie (2.2.1) pojawiło się w kwietniu 1997 r. Kolejne wydania z gałęzi 2.2 ujrzały światło dzienne w lecie i na jesieni 1997 r., przy czym ostatnie (2.2.8) pojawiło się w listopadzie 1998 r. Pierwsze oficjalne wydanie 3.0 pochodzi z października 1998 r. i stanowiło początek końca gałęzi 2.2.

Drzewo ewolucji FreeBSD ponownie rozdzieliło się 20 stycznia 1999 r., prowadząc do 4.0-CURRENT i 3.X-STABLE. Wersja 3.1 z 3.X-STABLE wydana została 15 lutego 1999, wersja 3.2 dnia 15 maja 1999, 3.3 w dniu 16 września 1999, 3.4 - 20 grudnia 1999 oraz 3.5 dnia 24 stycznia 2000. Wkrótce pojawiło się również pomniejsze wydanie 3.5.1, które zawierało kilka poprawek z ostatniej chwili do systemu Kerberos. Było to ostatnie wydanie gałęzi 3.X.

Kolejne rozgałęzienie miało miejsce 13 marca 2000 r. w wyniku czego pojawiła się gałąź 4.X-STABLE: 4.0-RELEASE w marcu 2000 i ostatnie wydanie 4.11-RELEASE w styczniu 2005.

Pojawienie się długo oczekiwanej gałęzi 5.0-RELEASE zostało ogłoszone 19 stycznia 2003 r. Stanowiła ona punkt kulminacyjny prawie trzyletniego wysiłku. Wydanie te wprowadziło FreeBSD na ścieżkę ku współpracy z komputerami multiprocessorowymi oraz zaawansowanej obsługi wątków aplikacji. Oferowała również wsparcie dla platform UltraSPARC i ia64. Wydanie 5.1 pojawiło się w czerwcu 2003 r. Ostatnie wydanie 5.X z gałęzi -CURRENT stanowiło 5.2.1-RELEASE, wprowadzone w lutym 2004.

Gałąź RELENG_5 powstała w sierpniu 2004 r., a także wydanie 5.3-RELEASE, stanowi początek wydań gałęzi 5-STABLE. Najnowsze wydanie 8.3-RELEASE pojawiło się w maju 2006. Wydawane będą wciąż kolejne wersje z gałęzi RELENG_5.

Kolejne rozgałęzienie nastąpiło w czerwcu 2005: powstała gałąź RELENG_6. Wydanie 6.0-RELEASE, pierwsze z

gałęzi 6.X, pojawiło się w listopadzie 2005. Najnowsze wydanie 9.1-RELEASE ujrzało światło dzienne w maju 2006 r. Będą pojawiać się również kolejne wydania z gałęzi RELENG_6.

Na chwilę obecną projekty długoterminowe prowadzone są w gałęzi 7.X-CURRENT. Migawki wydań 7.X, obrazujące postęp prac, są cały czas dostępne z serwera migawkowego ([ftp://current.FreeBSD.org/pub/FreeBSD/snapshots/](http://current.FreeBSD.org/pub/FreeBSD/snapshots/)) jak również na płytach CD.

1.3.2. Cele Projektu FreeBSD

Napisał Jordan Hubbard.

Głównym celem Projektu FreeBSD jest dostarczanie oprogramowania, które może być wykorzystane w dowolny sposób i bez dodatkowych zobowiązań. Wielu z nas ma duży wkład w tworzenie kodu (i rozwój projektu w ogóle) i z pewnością nie miałyby nic przeciw drobnemu wsparciu finansowemu. Tym nie mniej nie wywieramy żadnego nacisku. Wierzymy, że naszą pierwszą i najważniejszą „misją” jest dostarczanie kodu wszystkim tym, którzy go potrzebują bez względu na to do czego go wykorzystają, by zyskał on możliwie najszerszą bazę użytkowników dostarczając możliwie największych korzyści. W moim przekonaniu jest to jeden z najbardziej fundamentalnych celów stawianych przed całym Wolnym Oprogramowaniem, a przez nas entuzjastycznie wspierany.

Te części kodu w naszym drzewie źródłowym, które udostępniane są na licencji GNU General Public License (GPL) bądź Library General Public License (LGPL) posiadają kilka dodatkowych zobowiązań, choć związanych raczej z wymogiem udostępnienia kodu źródłowego. Z uwagi na dodatkowe komplikacje, które mogą pojawić się w przypadku komercyjnego zastosowania aplikacji na licencji GPL, osobiście skłaniamy się - kiedy jest to możliwe - ku oprogramowaniu dystrybuowanemu przy wykorzystaniu mniej restrykcyjnej licencji BSD.

1.3.3. Model rozwoju FreeBSD

Napisał Satoshi Asami.

Rzeczony FreeBSD jest otwartym i elastycznym procesem realizowanym przez setki ludzi na całym świecie (patrz Lista współpracowników (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/contributors/article.html)). Infrastruktura systemu rozwoju FreeBSD pozwala tymże setkom projektantów współpracować przez Internet. Tym nie mniej nieustannie poszukujemy nowych projektantów, a także nowych pomysłów. Osoby zainteresowane nawiązaniem bliższej współpracy z projektem mogą kontaktować się z nami bezpośrednio poprzez Techniczną listę dyskusyjną FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>). Natomiast Informacyjna lista dyskusyjna FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce>) jest również dostępna dla osób chcących poinformować innych użytkowników FreeBSD o głównych obszarach prowadzonych prac.

Oto garść informacji o projekcie FreeBSD i jego procesie rozwoju, przydatnych zarówno niezależnym projektantom jak i bliskim współpracownikom:

Repozytorium CVS

Główne drzewo źródłowe FreeBSD utrzymywane jest w systemie CVS (<http://ximbiot.com/cvs/wiki/>) (Concurrent Versions System) - wolnodostępnym narzędziu kontroli wersji kodu źródłowego, dostępnym we FreeBSD. Podstawowe repozytorium CVS (<http://www.FreeBSD.org/cgi/cvsweb.cgi>) znajduje się na maszynie zlokalizowanej w Santa Clara w Kalifornii, USA, skąd replikowane jest na serwery lustrzane, rozrzucone po całym świecie. Główne drzewo CVS, zawierające zarówno drzewo -CURRENT jak i -STABLE, można łatwo skopiować również na swój własny komputer. Proces ten został dokładniej opisany w podrozdziale Synchronizacja własnego drzewa kodu źródłowego.

Lista twórców

Twórcy są ludźmi, którzy posiadają prawa zapisu do drzewa CVS i posiadają upoważnienie do wprowadzania modyfikacji do kodu źródłowego FreeBSD. Angielski odpowiednik terminu „twórca” (ang. committer) pochodzi od polecenia `commit` systemu `cv`s(1), stosowanego do wprowadzania zmian do repozytorium CVS. Najlepszym sposobem przedstawienia własnych propozycji na liście dyskusyjnej twórców jest wykorzystanie polecenia `send-pr`(1). Jeśli system sprawia wrażenie zablokowanego można również wysłać e-mail bezpośrednio na Listę dyskusyjną twórców FreeBSD.

Główni projektanci FreeBSD

Porównując Projekt FreeBSD z przedsiębiorstwem, *zespół główny* należałoby porównać z zarządem firmy. Podstawowym zadaniem tejże grupy jest czuwanie nad prawidłowym rozwojem projektu jako całości. Jedną z funkcji grupy jest zapraszanie oddanych i odpowiedzialnych projektantów w szeregi twórców systemu, podobnie jak przyjmowanie w szeregi samej grupy. Obecna grupa została wybrana spośród wszystkich twórców w czerwcu 2004 r. Wybory mają miejsce co dwa lata.

Niektórzy z członków grupy posiadają również dodatkowy zakres obowiązków, tj. czuwają nad zapewnieniem poprawnego funkcjonowania wybranych części systemu. Pełna lista projektantów FreeBSD i ich obowiązków dostępna jest w artykule *Lista współpracowników* (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/contributors/article.html).

Notatka: Większość członków grupy jest ochotnikami, jeśli chodzi o rozwój FreeBSD, i nie otrzymują żadnego wynagrodzenia finansowego z projektu. Nie należy zatem błędnie interpretować „współpracy” z projektem jako „gwarancji wsparcia”. W tym świetle, powyższe porównanie z „zarządem” nie jest do końca celne. Bardziej odpowiednim byłoby powiedzieć, że są to ludzie, którzy z własnego wyboru oddali swój wolny czas dla FreeBSD!

Zewnętrzni współpracownicy

Co prawda jako ostatnia, ale zdecydowanie nie jako najmniej istotna, omówiona zostanie grupa współpracowników zewnętrznych, czyli samych użytkowników, którzy dostarczają na bieżąco informacji o funkcjonowaniu systemu oraz poprawek wykrytych błędów. Najlepszym sposobem na udział w rozwoju FreeBSD jest subskrypcja Technicznej listy dyskusyjnej FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>). Dodatek C zawiera więcej informacji o różnorodnych listach dyskusyjnych FreeBSD.

Lista współpracowników FreeBSD

(http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/contributors/article.html) cały czas rośnie. Czemu by nie dołączyć do listy pomagając w pracy nad FreeBSD już dzisiaj?

Pisanie kodu nie jest jedyną formą współpracy z projektem: kompletna lista rzeczy, które trzeba zrobić dostępna jest na stronie Projektu FreeBSD (<http://www.FreeBSD.org/index.html>).

Reasumując, nasz model rozwoju zorganizowany jest jako niezależne, współcentryczne okręgi. Scentralizowany model ma za zadanie ułatwić *użytkownikom* FreeBSD śledzenie zmian w kodzie. Odstraszanie potencjalnych współpracowników nie jest naszym celem! Pragniemy dostarczać stabilny system operacyjny z dużą bazą łatwych do instalacji i wykorzystania programów — ten model doskonale się w tym spisuje.

Jedynym o co prosimy tych, którzy mieliby wstąpić w szeregi projektantów FreeBSD, jest oddanie takie same jakie cechuje ich obecnych twórców.

1.3.4. Aktualne wydanie FreeBSD

FreeBSD jest łatwo dostępnym systemem operacyjnym, bazującym na kodzie 4.4BSD-Lite, dla następujących platform sprzętowych: Intel i386™, i486™, Pentium®, Pentium Pro, Celeron®, Pentium II, Pentium III, Pentium 4 (bądź inny zgodny), Xeon™, DEC Alpha oraz Sun UltraSPARC. Opiera się on przede wszystkim na oprogramowaniu grupy CSRG z Uniwersytetu Kalifornijskiego w Berkeley, rozszerzonym o dodatkowe elementy z NetBSD, OpenBSD, 386BSD i Free Software Foundation.

Począwszy od wydania FreeBSD 2.0 w końcu 1994 r., nastąpiła dramatyczna poprawa wydajności, możliwości i stabilności systemu. Największą zmianą była całkowita reformatacja systemu wirtualnej pamięci wraz ze współdzieloną pamięcią podręczną „VM/buffer cache”, która nie tylko wpłynęła na wzrost wydajności ale również zmniejszenie minimalnego miejsca zajmowanego w pamięci przez FreeBSD — 5 MB jest już akceptowalnym minimum. Inne rozszerzenia to m.in. kompletna obsługa klienta i serwera NIS, wsparcie dla transakcji TCP, wdzwanianie na żądanie PPP, zintegrowana obsługa DHCP, usprawniony podsystem SCSI, obsługa ISDN, ATM, FDDI, Fast i Gigabit Ethernet (100 i 1000 Mbit). Usprawniona obsługa najnowszych kontrolerów Adaptec i tysiące poprawionych błędów.

Oprócz podstawowej grupy aplikacji dystrybuowanych wraz z systemem, FreeBSD oferuje kolekcję tysięcy dodatkowych programów. W momencie pisania niniejszego tekstu ich lista obejmuje ponad 23,000 pozycji! Od serwerów http (WWW) poprzez gry po edytory i prawie wszystko pomiędzy. Cała Kolekcja Portów zajmuje około 500 MB na dysku, przy czym każdy port to zaledwie ułamek oryginalnej objętości źródeł. Takie rozwiązanie ułatwia aktualizację portów i zdecydowanie zmniejsza zajmowaną przestrzeń na dysku. Kompilacja portu sprowadza się do zmiany katalogu na zawierający port wybranego programu i wpisanie `make install`. Resztą zajmuje się system. Oryginalne pakiety źródeł dla każdego kompilowanego portu pobierane są dynamicznie z płyty CDROM bądź lokalnego serwera FTP. Wystarczy zadbać o dostateczną ilość wolnego miejsca na dysku. Dla osób nie mających ochoty kompilować programów własnoręcznie, większość portów jest również dostępna w skompilowanej postaci jako „pakiety”, które mogą być instalowane przy pomocy prostego polecenia `pkg_add`. Więcej informacji o systemie pakietów i portów zawiera Rozdział 4.

Dodatkowe dokumenty pomocne przy instalacji i użytkowaniu FreeBSD znajdują się również w katalogu `/usr/share/doc` na maszynach z najnowszymi wersjami FreeBSD. Mogą być przeglądane lokalnie za pomocą przeglądarki internetowej przy wykorzystaniu poniższych odnośników:

Podręcznik FreeBSD (ang.)

`/usr/share/doc/handbook/index.html`

FAQ FreeBSD (ang.)

`/usr/share/doc/faq/index.html`

Główne i najczęściej aktualizowane wersje dokumentów dostępne są na stronie <http://www.FreeBSD.org/>.

Rozdział 2. Instalacja FreeBSD

Rozdział przebudował i częściowo napisał od nowa Jim Mock. Omówienie sysinstall, zrzuty ekranów i inne fragmenty przygotował Randy Pratt. Tłumaczył Michał Wojciechowski.

2.1. Streszczenie

Wraz z FreeBSD rozpowszechniany jest prosty w użyciu program instalacyjny, działający w trybie tekstowym, o nazwie **sysinstall**. Jest on domyślnym programem instalacyjnym FreeBSD, jednakże dystrybutorzy systemu mogą zastąpić go własnym odpowiednikiem. W niniejszym rozdziale zawarto opis instalacji FreeBSD przy pomocy **sysinstall**.

Po przeczytaniu rozdziału będziemy wiedzieć:

- W jaki sposób tworzy się dyski instalacyjne FreeBSD.
- Jak FreeBSD odwołuje się do dysku i jak go dzieli.
- Jak uruchamia się **sysinstall**.
- Jakiek pytania zadaje **sysinstall**, o co w nich chodzi i jak na nie odpowiedzieć.

Przed przeczytaniem rozdziału powinniśmy:

- Zapoznać się z listą obsługiwanego sprzętu dołączoną do instalowanej wersji FreeBSD, by upewnić się, że posiadany sprzęt będzie działać.

Notatka: Opis instalacji dotyczy generalnie komputerów opartych na architekturze i386 („zgodny z PC”). W stosownych przypadkach podawane będą informacje odnoszące się do innych platform (na przykład Alpha). Pomimo starań o utrzymanie niniejszego opisu aktualnym, możliwe jest zaistnienie drobnych różnic pomiędzy instalatorem a zawartością tego rozdziału. Zaleca się, aby traktować niniejsze teksty jako ogólny przewodnik, niż raczej dosłowny podręcznik instalacji.

2.2. Czynności wstępne

Przekład uzupełnił Cezary Morga.

2.2.1. Rozpoznanie komponentów komputera

Przed instalacją FreeBSD powinniśmy zapoznać się z komponentami naszego komputera. W czasie instalacji FreeBSD pokaże listę urządzeń (dyski, karty sieciowe, napędy CD-ROM, itd.) wraz z informacjami o producentach i numerach modeli. FreeBSD postara się także ustalić prawidłową konfigurację każdego z nich, m.in. ustawienia przerwań IRQ i portów we/wy. Ze względu na „kaprysy” pcetowego sprzętu może się okazać, że konfiguracja wykryta przez FreeBSD nie jest w pełni prawidłowa i trzeba będzie samodzielnie ją poprawić.

Jeżeli na komputerze jest już zainstalowany inny system operacyjny, na przykład Windows lub Linux, warto jest skorzystać z dostępnych w nim narzędzi do sprawdzenia bieżącej konfiguracji sprzętowej. Kiedy zupełnie nie

wiadomo jak skonfigurowana powinna być dana karta, wymagane informacje mogą znajdować się bezpośrednio na niej samej. Często spotykane numery przerwań IRQ to 3, 5 i 7, a adresy portów we/wy są zwykle zapisywane w postaci liczb szesnastkowych, na przykład 0x330.

Zalecamy by zebrane informacje wydrukować lub zapisać na kartce przed rozpoczęciem instalacji FreeBSD. Można je zestawić w postaci tabeli, np.:

Tabela 2-1. Przykładowa lista urządzeń

Nazwa urządzenia	IRQ	Port(y) we/wy	Uwagi
Pierwszy dysk twardy	brak	brak	40 GB, firmy Seagate, IDE 1 master
CDROM	brak	brak	IDE 1 slave
Drugi dysk twardy	brak	brak	20 GB, firmy IBM, IDE 2 master
Kontroler IDE	14	0x1f0	
Karta sieciowa	brak	brak	Intel® 10/100
Modem	brak	brak	3Com® 56K faxmodem na COM1
...			

2.2.2. Przygotowanie kopii danych

Jeśli komputer, na którym będzie przeprowadzana instalacja zawiera cenne dane, powinniśmy koniecznie przygotować ich kopię zapasową, oraz sprawdzić stan tychże kopii przed instalacją FreeBSD. Podczas instalacji kilkakrotnie pojawi się prośba o potwierdzenie przed zapisaniem czegokolwiek na dysku, jednak gdy już się to rozpocznie, nie będzie możliwości odwrotu.

2.2.3. Wybór miejsca dla FreeBSD

Jeżeli masz zamiar przeznaczyć cały dysk na FreeBSD, to omawiane poniżej zagadnienia nie będą cię dotyczyć — możesz pominąć tę część.

W przypadku, gdy zamierzamy zainstalować FreeBSD obok innych systemów operacyjnych, warto zapoznać się z podstawowymi informacjami o sposobie przechowywania danych na dysku.

2.2.3.1. Układ dysku w systemach i386™

Dysk komputera typu PC można podzielić na oddzielne porcje, zwane *partycjami*. Komputery PC potrafią obsłużyć maksymalnie cztery partycje na jednym dysku. Partycje te nazywane są *partycjami podstawowymi*. W celu ominięcia tego ograniczenia i umożliwienia stworzenia większej liczby partycji, wymyślono nowy typ partycji - *partycje rozszerzone*. Na dysku może znajdować się tylko jedna taka partycja. Natomiast wewnątrz niej można utworzyć specjalne partycje, zwane *partycjami logicznymi*.

Wszystkie partycje posiadają własny *identyfikator partycji*, tj. numer określający typ przechowywanych na niej danych. Partycje FreeBSD oznaczone są identyfikatorem 165.

Każdy ze stosowanych systemów operacyjnych identyfikuje partycje w określony sposób. Dla przykładu, DOS i jego następcy, w tym Windows, przypisują każdej partycji podstawowej i logicznej *literę dysku*, zaczynając od C:.

FreeBSD musi być zainstalowane na partycji podstawowej. Wszystkie własne dane, w tym pliki tworzone przez użytkowników, może przechowywać na jednej partycji. Jednakże, jeśli masz do dyspozycji kilka dysków, możesz utworzyć partycję FreeBSD na każdym z nich bądź jedynie na wybranych. Tym nie mniej na potrzeb instalacji wymagane jest posiadanie jednej partycji. Może to być świeżo utworzona, pusta partycja, lub też partycja zawierająca dane, które nie są już potrzebne.

W przypadku, gdy wszystkie dostępne partycje na dysku są już wykorzystywane, będziesz musiał zwolnić jedną z nich, korzystając z narzędzi dostępnych w wykorzystywanym systemie operacyjnym (np. `fdisk` w DOS lub Windows).

Jeśli dysponujesz wolną partycją, możesz ją wykorzystać. Może się jednak okazać, że zajdzie potrzeba zmniejszenia rozmiarów niektórych z pozostałych partycji.

Minimalna instalacja FreeBSD zajmuje jedynie 100 MB miejsca na dysku. Jest to jednakże *bardzo* minimalna instalacja, praktycznie nie pozostawiająca miejsca na pliki użytkowników. Zdecydowanie bardziej realnym minimum jest 250 MB, o ile nie planujemy wykorzystania środowiska graficznego, bądź co najmniej 350 MB z graficznym interfejsem. Instalowanie wielu dodatkowych programów wymaga więcej wolnego miejsca na dysku.

W celu przygotowania miejsca dla FreeBSD można wykorzystać narzędzia komercyjne pokroju **PartitionMagic®** bądź darmowe jak **GParted**. Dwa darmowe programy służące do tego samego celu, tj. **FIPS** i **PResizer**, dostępne są na płycie CD w katalogu `tools`. W tym samym katalogu znajduje się również ich dokumentacja. Zarówno **FIPS**, **PResizer** jak i **PartitionMagic** potrafią rozszerzać partycje typu FAT16 i FAT32 — wykorzystywane w MS-DOS aż po Windows ME. System plików NTFS potrafią obsługiwać **PartitionMagic** i **GParted**.

Ostrzeżenie Niewłaściwe korzystanie z tych narzędzi może doprowadzić do utraty danych. Przed ich zastosowaniem należy się upewnić, że przygotowaliśmy aktualne kopie zapasowe.

Przykład 2-1. Wykorzystanie niezmienionej istniejącej partycji

Przyjmijmy, że mamy do dyspozycji komputer wyposażony w dysk o pojemności 4 GB, z zainstalowanym systemem Windows. Dysk jest podzielony na dwie części oznaczone literami `C:` i `D:`, o rozmiarze 2 GB każda. Na `C:` mamy 1 GB danych, a na `D:` 0,5 GB danych.

Mamy więc dysk o dwóch partycjach, z których każda oznaczona jest literą dysku. Możemy skopiować dane z `D:` na `C:`, dzięki czemu druga partycja stanie się wolna i będzie można zainstalować na niej FreeBSD.

Przykład 2-2. Zmniejszenie istniejącej partycji

Przyjmijmy tym razem, że na dysku o pojemności 4 GB zainstalowany jest system Windows na jednej dużej partycji. Partycja dostępna jest jako dysk `C:` o rozmiarze 4 GB. Mamy na nim 1,5 GB danych i chcielibyśmy udostępnić dla FreeBSD 2 GB.

Możemy wybrać jedno z poniższych rozwiązań:

1. Przygotować kopię danych, następnie na nowo zainstalować Windows, tworząc podczas instalacji partycję o rozmiarze 2 GB.
2. Skorzystać z jednego ze wspomnianych wcześniej narzędzi, np. **PartitionMagic**, w celu zmniejszenia rozmiaru partycji Windows.

2.2.3.2. Układ dysku Alpha

W przypadku architektury Alpha na FreeBSD trzeba będzie przeznaczyć cały dysk. Nie ma obecnie możliwości wspólnego korzystania z dysku przez kilka systemów operacyjnych. W zależności od konkretnego modelu komputera Alpha, możemy wykorzystać dysk SCSI lub IDE, o ile komputer umożliwia załadowanie z niego systemu operacyjnego.

Zgodnie z konwencją stosowaną w podręcznikach Digital / Compaq wszystkie polecenia SRM pisane są wielkimi literami. SRM nie rozróżnia małych i dużych liter.

By wyświetlić nazwy i rodzaje zainstalowanych w komputerze dysków, posługujemy się poleceniem `SHOW DEVICE` w konsoli SRM:

```
>>>SHOW DEVICE
dka0.0.0.4.0          DKA0          TOSHIBA CD-ROM XM-57  3476
dkc0.0.0.1009.0       DKC0          RZ1BB-BS  0658
dkc100.1.0.1009.0     DKC100        SEAGATE ST34501W  0015
dva0.0.0.0.1          DVA0
ewa0.0.0.3.0          EWA0          00-00-F8-75-6D-01
pkc0.7.0.1009.0       PKC0          SCSI Bus ID 7  5.27
pqa0.0.0.4.0          PQA0          PCI EIDE
pqb0.0.1.4.0          PQB0          PCI EIDE
```

Powyższy przykład pochodzi z komputera Digital Personal Workstation 433au i pokazuje trzy dyski. Pierwszym z nich jest CDROM opisany nazwą `DKA0`, natomiast dwa pozostałe to twarde dyski o nazwach `DKC0` i `DKC100`.

Dyski o nazwach typu `DKx` są dyskami SCSI. Dla przykładu `DKA100` oznacza dysk SCSI o identyfikatorze 1 na pierwszej szynie SCSI (A), natomiast `DKC300` oznacza dysk o identyfikatorze 3 na trzeciej szynie SCSI (C). Nazwa `PKx` oznacza kontroler SCSI. Jak pokazuje przykład z `SHOW DEVICE`, napędy CDROM SCSI traktowane są tak samo jak dyski twarde SCSI.

Nazwy dysków IDE mają postać `DQx`, a nazwa `PQx` oznacza kontroler IDE.

2.2.4. Zbieranie informacji o konfiguracji sieci

Jeśli podczas instalacji będziemy korzystać z połączenia z siecią (np. FreeBSD instalowane będzie z serwera FTP lub serwera NFS), będziemy musieli znać konfigurację sieci. W trakcie instalacji pojawi się prośba o wpisanie tej konfiguracji, by umożliwić FreeBSD połączenie się z siecią i kontynuowanie instalacji.

2.2.4.1. Połączenie z siecią Ethernet lub przez modem kablowy/DSL

W przypadku komputera podłączonego do sieci Ethernet lub połączonego z Internetem przez modem kablowy lub DSL, potrzebne będą następujące informacje:

1. Adres IP
2. Adres IP domyślnej bramy
3. Nazwa stacji

4. Adresy IP serwerów DNS
5. Maska podsieci

Informacje te możemy uzyskać od administratora systemu lub dostawcy usług sieciowych. Może się okazać, że konfiguracja odbywa się automatycznie, przy użyciu *DHCP*. Jeśli tak jest, należy o tym fakcie pamiętać.

2.2.4.2. Połączenie przez modem

Instalacja FreeBSD przez Internet możliwa jest także w przypadku połączenia modemowego, jednakże będzie to trwało bardzo długo.

Niezbędne informacje:

1. Numer telefonu do dostawcy usług internetowych
2. Numer portu COM, do którego podłączony jest modem
3. Nazwa użytkownika i hasło konta u dostawcy usług

2.2.5. Sprawdzenie erraty FreeBSD

W pracy nad FreeBSD podejmowane są wszelkie starania, aby każde wydanie FreeBSD było jak najbardziej niezawodne, jednakże od czasu do czasu zdarzają się błędy. W pewnych bardzo rzadkich przypadkach mogą mieć one wpływ na proces instalacji systemu. Błędy te po wykryciu i naprawieniu są opisywane w erracie zamieszczonej na stronie FreeBSD Errata (<http://www.FreeBSD.org/releases/9.1R/errata.html>) (ang.). Przed instalacją warto jest sprawdzić, czy w erracie nie wspomniano o problemach, które mogą zakłócić instalację.

Informacje o wszystkich wydaniach systemu, jak również erraty do każdego z nich, znaleźć można na stronie WWW FreeBSD (<http://www.FreeBSD.org/index.html>) w części poświęconej wydaniom (<http://www.FreeBSD.org/releases/index.html>).

2.2.6. Pozyskanie plików instalacyjnych FreeBSD

Pliki potrzebne do rozpoczęcia instalacji systemu mogą pochodzić z jednego z wymienionych poniżej źródeł:

Nośniki lokalne

- Płyta CDROM lub DVD
- Partycja DOS-owa na tym samym komputerze
- Pamięć taśmowa QIC lub SCSI
- Dyskietki

Sieć

- Serwer FTP, także przez firewall lub proxy HTTP, zależnie od potrzeb
- Serwer NFS
- Dedykowane połączenie równoległe lub szeregowo

Posiadając FreeBSD na CD lub DVD, mamy już wszystko, co potrzeba, możemy zatem przejść do następnej części (Sekcja 2.2.7).

Jeśli nie mamy plików instalacyjnych FreeBSD, Sekcja 2.13 zawiera opis instalacji FreeBSD z dowolnego z wymienionych wcześniej źródeł. Następnie powróćmy do Sekcja 2.2.7.

2.2.7. Przygotowanie dyskietek do instalacji

Instalacja FreeBSD rozpoczyna się uruchomieniem programu instalacyjnego podczas startu komputera — nie jest to program, który można uruchomić w innym systemie operacyjnym. Zwykle przy uruchamianiu komputera ładowany jest system zainstalowany na dysku twardym, jednak można także uruchomić system z dyskietki „startowej”. Do tego celu może także posłużyć CDROM, jeśli komputer daje taką możliwość.

Podpowiedź: Jeśli posiadamy FreeBSD na płytach CDROM lub DVD (kupionych lub przygotowanych samodzielnie), a nasz komputer pozwala na uruchomienie z płyty (zwykle dzięki ustawieniu opcji BIOS-u zwanej „Boot Order” lub podobnej), możemy nie czytać niniejszej części. Płyty CDROM i DVD zawierające FreeBSD mogą być użyte jako dyski startowe bez dodatkowego przygotowania.

By utworzyć zestaw dyskietek startowych, należy:

1. Zdobyć obrazy dyskietek startowych

Dyskietki startowe znaleźć można wśród plików instalacyjnych w katalogu `floppies/` bądź pobrać z serwera `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/<arch>/<version>-RELEASE/floppies/` zamieniając odpowiednio `<arch>` i `<wersja>` właściwą architekturą naszego sprzętu i wybraną wersją FreeBSD. Przykładowo, obrazy dyskietek dla FreeBSD 9.1-RELEASE na architekturę i386 dostępne są pod adresem `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/9.1-RELEASE/floppies/`.

Obrazy dyskietek mają rozszerzenie `.flp`. Katalog `floppies/` zawiera kilka różnych obrazów, a to, które z nich będą potrzebne, zależy od wersji FreeBSD, która będzie instalowana, a czasem również od sprzętu na którym system ma być zainstalowany. Z reguły potrzebne będą trzy dyskietki `boot.flp`, `kern1.flp` i `kern2.flp`. Warto jednak dla pewności przeczytać znajdujący się w tym samym katalogu plik `README.TXT`.

Notatka: Systemy gałęzi 5.X starsze od FreeBSD 5.3 mogą wymagać dodatkowych sterowników urządzeń. Znaleźć je można w obrazie dyskietki `drivers.flp`.

WAŻNE: Pobierając pliki przez FTP należy koniecznie używać trybu binarnego. Wiadomo jest, że w niektórych przeglądarkach stosowany jest tryb tekstowy (zwany też ASCII), przez co dyskietki startowe mogą się okazać niezdдатne do użycia.

2. Przygotować dyskietki startowe

Dla każdego pliku z obrazem przygotowujemy jedną dyskietkę. Dyskietki nie mogą być w jakikolwiek sposób uszkodzone. Najprostszym sposobem samodzielnego sprawdzenia, czy dyskietka nie jest wadliwa, jest jej sformatowanie. Nie powinniśmy ufać dyskietkom formatowanym fabrycznie. Narzędzie formatujące dostępne w

systemie Windows nie poinformuje o istnieniu uszkodzonych bloków, po prostu oznaczy je jako „uszkodzone” i zignoruje. Zaleca się używanie fabrycznie nowych dyskietek.

WAŻNE: Gdy podczas instalacji FreeBSD program instalacyjny wskaże błąd, zastygnie lub zachowa się w dziwny sposób, jednymi z pierwszych podejrzanych powinny być dyskietki. Trzeba wówczas nagrać pliki obrazów na inne dyskietki i spróbować ponownie.

3. Nagrać pliki obrazów na dyskietki

Pliki `.flp` nie są zwyczajnymi plikami, które można nagrać na dyskietkę. Są natomiast obrazami całkowitej zawartości dyskietek. Oznacza to, że *nie można* zapisać tych plików po prostu kopiując z jednego dysku na drugi. Skorzystamy ze specjalnego oprogramowania, by bezpośrednio zapisać obrazy na dyskietkach.

Jeśli dyskietki nagrywamy na komputerze z MS-DOS/Windows, to możemy skorzystać z dołączonego do FreeBSD narzędzia `fdimage`.

W przypadku, gdy wykorzystujemy obrazy dyskietek z płyty CDROM dostępnego jako dysk `E:`, posłużymy się poleceniem:

```
E:\> tools\fdimage floppies\kern.flp A:
```

Powtarzamy je dla każdego z plików `.flp`, za każdym razem zmieniając dyskietkę. Najlepiej jest też napisać na dyskietce nazwę skopiowanego na nią pliku. Powyższe polecenie może potrzebować pewnych modyfikacji, w zależności od miejsca, w którym znajdują się pliki `.flp`. Jeżeli nie dysponujemy płytą CD, możemy pobrać `fdimage` z katalogu `tools` (<ftp://ftp.FreeBSD.org/pub/FreeBSD/tools/>) na serwerze FTP FreeBSD.

Jeżeli natomiast dyskietki nagrywamy w systemie uniksowym (na przykład w innym FreeBSD), do zapisania plików obrazów na dyskietkach możemy wykorzystać polecenie `dd(1)`. We FreeBSD wpisalibyśmy:

```
# dd if=kern.flp of=/dev/fd0
```

W systemie FreeBSD `/dev/fd0` odpowiada pierwszej stacji dyskietek (napędowi `A:`). `/dev/fd1` odpowiadałoby `B:` i tak dalej. W innych odmianach systemów UNIX mogą być stosowane inne nazwy stacji dyskietek, konieczne może więc być zapoznanie się z dokumentacją danego systemu.

W tej chwili jesteśmy już przygotowani do instalacji FreeBSD.

2.3. Rozpoczęcie instalacji

WAŻNE: Z założenia, podczas instalacji dane na dysku (lub dyskach) nie ulegną żadnym zmianom przed pojawieniem się następującego komunikatu:

```
Last Chance: Are you SURE you want continue the installation?
```

```
If you're running this on a disk with data you wish to save then WE
STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!
```

```
We can take no responsibility for lost disk contents!
```

Instalację można przerwać w dowolnej chwili przed powyższym ostrzeżeniem, mając pewność, że dane na dysku pozostają nietknięte. Jeśli będziemy się obawiać, że coś niewłaściwie skonfigurowaliśmy, możemy po prostu wyłączyć komputer i nic złego się nie stanie.

2.3.1. Uruchomienie komputera

2.3.1.1. Uruchomienie i386

1. Na początku komputer powinien być wyłączony.
2. Włączamy komputer. Po chwili powinna pojawić się możliwość przejścia do menu systemowego, lub BIOS-u, najczęściej poprzez naciśnięcie klawisza **F2**, **F10**, **Del** bądź **Alt+S**. Wciskamy odpowiedni klawisz zgodnie z informacją na ekranie. Niekiedy komputer podczas uruchamiania pokazuje jakiś obrazek. Zwykle wciskając **Esc** możemy pozbyć się obrazka, aby mieć możliwość przeczytania komunikatów.
3. Wśród opcji odnajdujemy tę, która decyduje o kolejności ładowania systemu z poszczególnych urządzeń. Zwykle ma ona postać listy urządzeń, takich jak `Floppy`, `CDROM`, `First Hard Disk`, itd.

Jeżeli wcześniej przygotowaliśmy dyskietki startowe, wybieramy stację dyskietek. Jeśli natomiast korzystamy z płyty CD, wybieramy właśnie `CDROM`. Wątpliwości możemy rozstrzygnąć zaglądając do instrukcji dołączonej do komputera i jego płyty głównej.

Wprowadzone zmiany muszą być zapisane przed opuszczeniem menu systemowego. Komputer powinien ponownie się uruchomić.

4. Jeżeli korzystamy z dyskietek startowych, o których traktuje Sekcja 2.2.7, to jedna z nich będzie pierwszą dyskietką startową, najprawdopodobniej będzie to dyskietka zawierająca `kern.flp`. Ją właśnie wkładamy do stacji.

W przypadku korzystania z płyty CD wystarczy po prostu włączyć komputer i włożyć płytę do napędu.

Jeżeli komputer uruchomi się jak zwykle i załadowuje już zainstalowany system operacyjny, może to oznaczać, że:

1. Dyskietka lub płyta zostały włożone za późno. Powinniśmy spróbować uruchomić komputer bez wyjmowania dyskietki bądź płyty.
 2. Zmiany w ustawieniach BIOS-u nie zadziałały prawidłowo. Spróbujmy wprowadzić je ponownie, aż do osiągnięcia zamierzonego efektu.
 3. Nasza wersja BIOS-u nie pozwala na uruchomienie systemu z wybranego nośnika.
5. Rozpocznie się ładowanie FreeBSD. Podczas ładowania z płyty CD pojawi się tekst podobny do poniższego (pominięto informacje o wersji)::

```
Verifying DMI Pool Data .....
Boot from ATAPI CD-ROM :
  1. FD 2.88MB  System Type-(00)
Uncompressing ... done

BTX loader 1.00 BTX version is 1.01
Console: internal video/keyboard
BIOS drive A: is disk0
BIOS drive B: is disk1
BIOS drive C: is disk2
BIOS drive D: is disk3
BIOS 639kB/261120kB available memory
```

```
FreeBSD/i386 bootstrap loader, Revision 0.8
```

```
/kernel text=0x277391 data=0x3268c+0x332a8 |
```

```
|
Hit [Enter] to boot immediately, or any other key for command prompt.
Booting [kernel] in 9 seconds... _
```

Natomiast ładując z dyskietki, zobaczymy tekst w rodzaju (pominięto informacje o wersji):

```
Verifying DMI Pool Data .....
```

```
BTX loader 1.00  BTX version is 1.01
Console: internal video/keyboard
BIOS drive A: is disk0
BIOS drive C: is disk1
BIOS 639kB/261120kB available memory
```

```
FreeBSD/i386 bootstrap loader, Revision 0.8
```

```
/kernel text=0x277391 data=0x3268c+0x332a8 |
```

```
Please insert MFS root floppy and press enter:
```

Postępując zgodnie z instrukcją na ekranie, wyjmujemy dyskietkę `kern.flp`, wkładamy `mfsroot.flp` i naciskamy **Enter**. We FreeBSD 5.3 i późniejszych dostępne są również inne dyskietki opisane w poprzednim podrozdziale. Należy uruchomić system z pierwszej dyskietki, następnie wkładać kolejne zgodnie z pojawiającymi się komunikatami.

6. Niezależnie, czy uruchamiamy komputer z dyskietki czy z płyty, podczas ładowania ujrzymy komunikat:

```
Hit [Enter] to boot immediately, or any other key for command prompt.
Booting [kernel] in 9 seconds... _
```

Albo czekamy dziesięć sekund, albo wciskamy **Enter**.

2.3.1.2. Uruchomienie Alpha

1. Na początku komputer powinien być wyłączony.
2. Włączamy komputer i czekamy na znak zachęty boot monitora.
3. Jeżeli korzystamy z dyskietek startowych opisanych w Sekcja 2.2.7, to jedna z nich będzie pierwszą dyskietką startową, najprawdopodobniej będzie to dyskietka zawierająca `kern.flp`. Ją właśnie wkładamy do stacji i wpisujemy następujące polecenie, aby uruchomić komputer z dyskietki (zmieniając nazwę napędu dyskietek, jeżeli będzie to konieczne):

```
>>>BOOT DVA0 -FLAGS " -FILE "
```

W przypadku korzystania z płyty CD, wkładamy ją do napędu i rozpoczynamy instalację wpisując następujące polecenie (wstawiając inną nazwę napędu CDROM, jeżeli będzie to konieczne):

```
>>>BOOT DKA0 -FLAGS " -FILE "
```

4. Rozpocznie się ładowanie FreeBSD. Podczas ładowania z dyskietki, zobaczymy tekst w rodzaju:

```
Please insert MFS root floppy and press enter:
```

Postępując zgodnie z instrukcją na ekranie, wyjmujemy dyskietkę `kern.flp`, wkładamy `mfsroot.flp` i naciskamy **Enter**.

5. Niezależnie, czy uruchamiamy komputer z dyskietki czy z płyty, podczas ładowania ujrzymy komunikat:

```
Hit [Enter] to boot immediately, or any other key for command prompt.
Booting [kernel] in 9 seconds... _
```

Czekamy dziesięć sekund, albo wciskamy **Enter**. Przejdziemy do menu konfiguracyjnego jądra.

2.3.2. Przeglądanie wyników rozpoznania urządzeń

Kilkaset ostatnio wyświetlonych na ekranie linii jest zapisywanych i można je przeglądać.

By przejrzeć bufor, naciskamy **Scroll Lock**. Włączamy w ten sposób tryb przewijania ekranu. Można teraz przeglądać wyniki rozpoznania urządzeń przy użyciu klawiszy kursora, lub **PageUp** i **PageDown**. Tryb przewijania wyłącza się wciskając ponownie **Scroll Lock**.

Zróbmy to, aby przejrzeć tekst, który został przewinięty poza ekran, gdy jądro dokonywało rozpoznawania urządzeń. Tekst będzie mieć treść podobną do przedstawionej na Rysunek 2-1, jednakże dokładna treść zależy od zainstalowanych w komputerze urządzeń.

Rysunek 2-1. Przykład wyników rozpoznania urządzeń

```
avail memory = 253050880 (247120K bytes)
Preloaded elf kernel "kernel" at 0xc0817000.
Preloaded mfs_root "/mfsroot" at 0xc0817084.
md0: Preloaded image </mfsroot> 4423680 bytes at 0xc03ddcd4

mdl: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1: <VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <ISA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0 <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci
0
usb0: <VIA 83572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr1
uhub0: 2 ports with 2 removable, self powered
pci0: <unknown card> (vendor=0x1106, dev=0x3040) at 7.3
dc0: <ADMtek AN985 10/100BaseTX> port 0xe800-0xe8ff mem 0xdb000000-0xeb0003ff ir
q 11 at device 8.0 on pci0
```

```

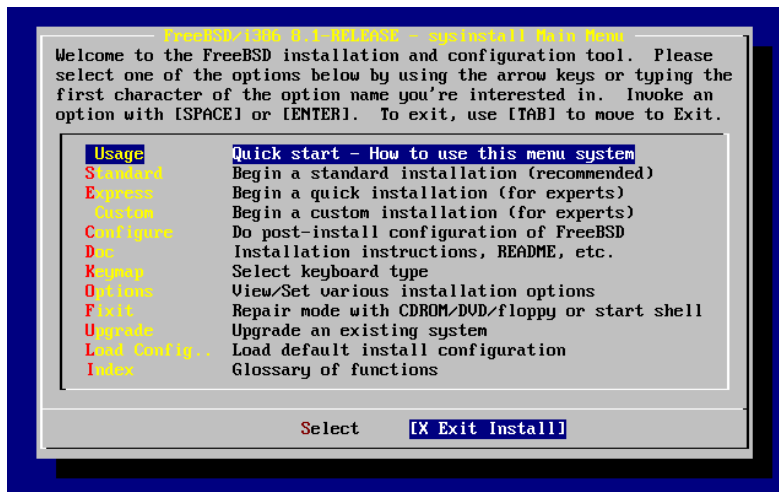
dc0: Ethernet address: 00:04:5a:74:6b:b5
miibus0: <MII bus> on dc0
ukphy0: <Generic IEEE 802.3u media interface> on miibus0
ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xec00-0xec1f irq 9 at device 10.
0 on pci0
ed0 address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
orm0: <Option ROM> at iomem 0xc0000-0xc7fff on isa0
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/@ mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
pppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
plip0: <PLIP network interface> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master UDMA33
acd0: CD-RW <LITE-ON LTR-1210B> at ata1-slave PIO4
Mounting root from ufs:/dev/md0c
/stand/sysinstall running as init on vty0

```

Warto jest uważnie przejrzeć wyniki, by mieć pewność, że wszystkie spodziewane urządzenia zostały wykryte. Brak urządzenia na liście oznacza, że nie zostało ono wykryte. Jeśli sterownik wymagał skonfigurowania IRQ i adresu portu, to powinniśmy sprawdzić, czy prawidłowo je wpisaliśmy.

Jeśli trzeba będzie zmienić ustawienia rozpoznawania urządzeń, możemy łatwo opuścić program **sysinstall** i zacząć od nowa. Dzięki temu można również lepiej poznać cały proces.

Rysunek 2-2. Wyjście z sysinstall



Korzystając z klawiszy kursora, wybieramy z głównego menu **Exit Install**. Ukaze się następujący komunikat:

```

User Confirmation Requested
Are you sure you wish to exit? The system will reboot
(be sure to remove any floppies from the drives).

[ Yes ]    No

```

Instalacja ponownie zacznie się od początku, jeśli wybierzemy **[Yes]**, pozostawiając płytę CD w napędzie.

Jeśli instalujemy z dyskietek, przed ponownym uruchomieniem komputera powinniśmy wyjąć dyskietkę `mfsroot.flp` i włożyć `kern.flp`.

2.4. Wprowadzenie do sysinstall

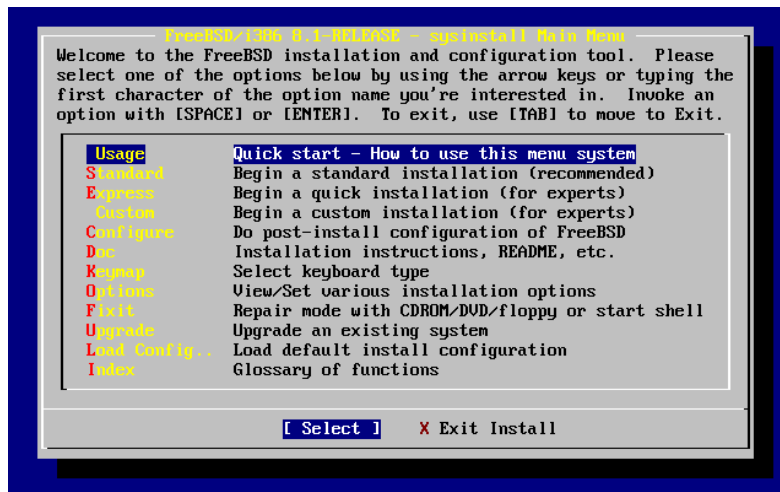
Sysinstall jest aplikacją instalacyjną przygotowaną w ramach Projektu FreeBSD. Jest to program konsolowy podzielony na szereg pomniejszych menu i ekranów, służących do konfiguracji i zarządzania procesem instalacji.

Menu **sysinstall** obsługiwane jest klawiszami kursora, klawiszem **Enter**, **Spacją** i innymi. Dokładny opis działania poszczególnych klawiszy znaleźć można w części poświęconej posługiwaniu się **sysinstall**.

Dostęp do tych informacji możliwy jest poprzez podświetlenie pozycji **Usage** i wybranie przycisku **[Select]**, a następnie wciśnięcie klawisza **Enter**, zgodnie z Rysunek 2-3.

Wyświetlone zostaną wskazówki odnośnie posługiwania się systemem menu. Po ich przeczytaniu powrót do głównego menu możliwy jest poprzez naciśnięcie klawisza **Enter**.

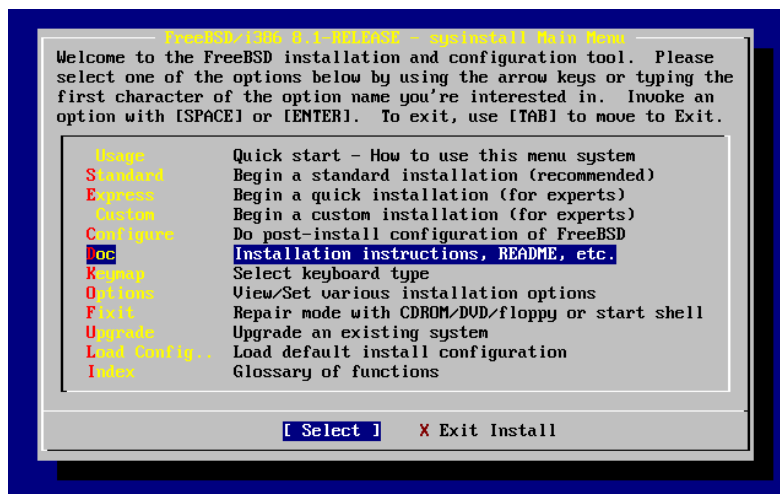
Rysunek 2-3. Wyświetlenie z głównego menu instrukcji obsługi sysinstall



2.4.1. Menu dokumentacji

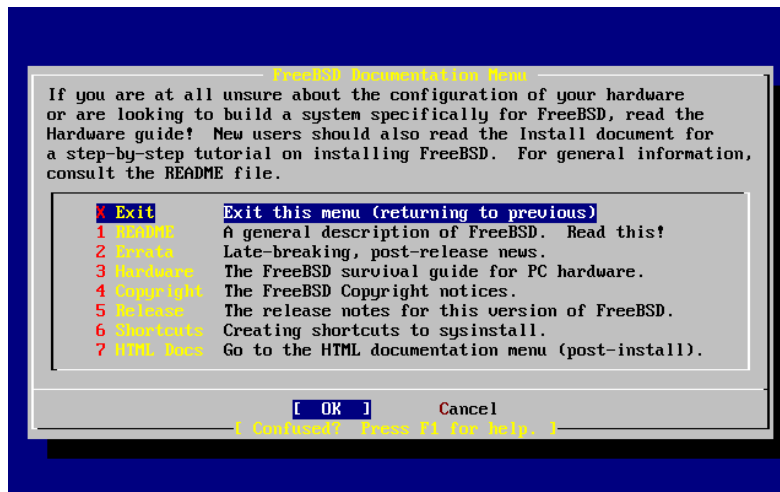
Korzystając z klawiszy kursora, w głównym menu wybieramy Doc i wciskamy **Enter**.

Rysunek 2-4. Wybór menu dokumentacji



Spowoduje to wyświetlenie menu dokumentacji.

Rysunek 2-5. Menu dokumentacji sysinstall



Warto przeczytać dostępne tu dokumenty.

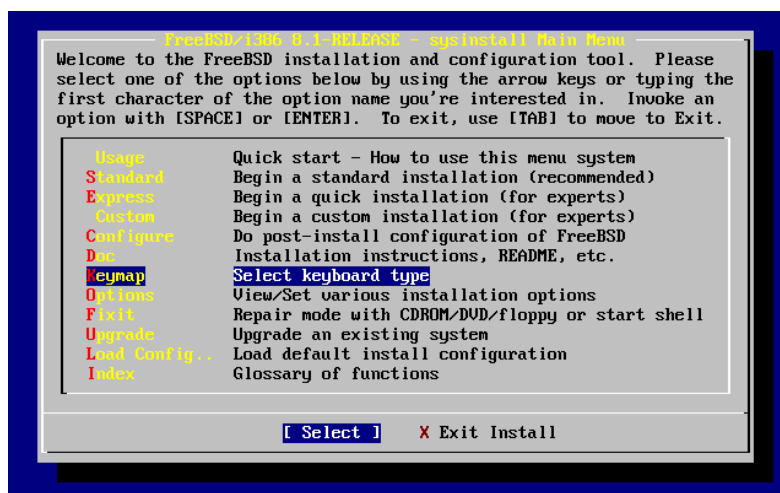
By wyświetlić konkretny dokument, wybieramy go klawiszami kursora, a następnie wciskamy **Enter**. Po przeczytaniu klawiszem **Enter** możemy powrócić do menu dokumentacji.

Do głównego menu instalacji powracamy wybierając klawiszami kursora **Exit**, a następnie wciskając **Enter**.

2.4.2. Menu mapowania klawiatury

Aby zmienić mapowanie klawiatury klawiszami kursora wybieramy z menu pozycję **Keymap** i wciskamy **Enter**. Zmiana mapowania klawiatury wymagana jest jedynie gdy używamy klawiatury innej niż standardowej amerykańskiej.

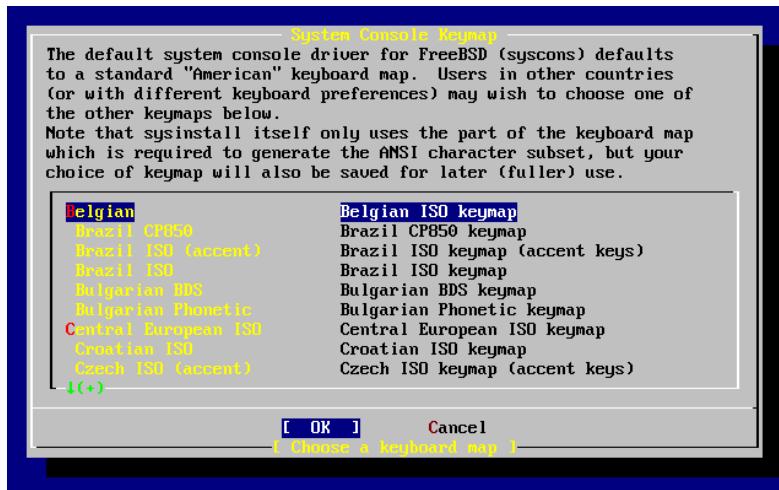
Rysunek 2-6. Główne menu sysinstall



Wyboru mapowania klawiatury dokonujemy poprzez wskazanie odpowiedniej pozycji z listy przy pomocy klawiszy kursora, oraz wciśnięcie **Spacji**. Ponowne naciśnięcie **Spacji** cofa wybór. Po wybraniu odpowiedniego mapowania wskazujemy klawiszami kursora [OK] i wciskamy **Enter**.

Na poniższym rysunku przedstawiona jest tylko część listy. Wybranie [Cancel] spowoduje przyjęcie domyślnego mapowania klawiatury i powrót do głównego menu.

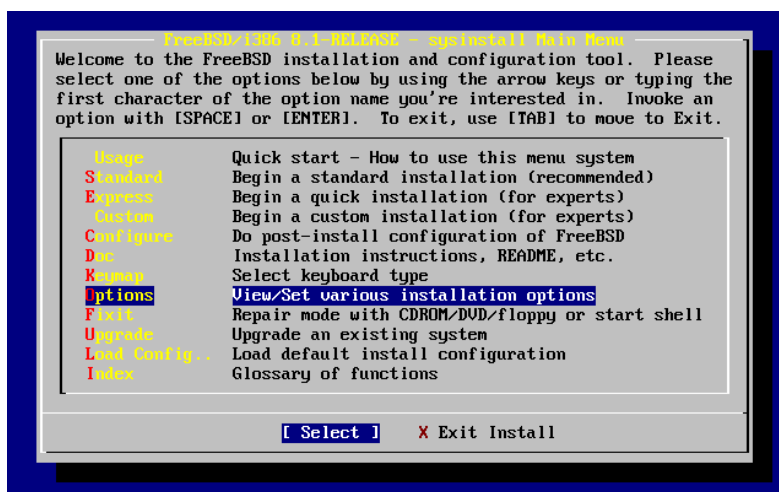
Rysunek 2-7. Menu mapowania klawiatury



2.4.3. Ekran opcji instalacji

Wybieramy Options i naciskamy **Enter**.

Rysunek 2-8. Główne menu sysinstall



Rysunek 2-9. Opcje sysinstall

```
Options Editor
Name      Value      Name      Value
-----
NFS Secure NO          Browser Exec /usr/local/bin/links
NFS Slow  NO          Media Type  <not yet set>
NFS TCP   NO          Media Timeout 300
NFS version 3 YES       Package Temp /var/tmp
Debugging NO          Newfs Args -b 16384 -f 2048
No Warnings NO       Fixit Console serial
Yes to All NO       Re-scan Devices <*>
DHCP      NO          Use Defaults [RESET!]
IPv6      NO
FTP username ftp
Editor    /usr/bin/ee
Extract Detail high
Release Name 8.1-RELEASE
Install Root /
Browser package links

Use SPACE to select/toggle an option, arrow keys to move,
? or F1 for more help. When you're done, type Q to Quit.

NFS server talks only on a secure port
```

Wartości domyślne są zwykle odpowiednie dla większości użytkowników i nie ma potrzeby ich zmiany. Nazwa wydania może być inna w zależności od instalowanej wersji systemu.

Po wybraniu jednej z opcji, na dole ekranu ukaże się jej opis podświetlony na niebiesko. Opcja Use Defaults (użyj domyślnych) przywraca wszystkim opcjom wartości domyślne.

Naciskając **F1** przechodzimy do ekranu pomocy, gdzie możemy przeczytać o poszczególnych opcjach.

Naciskając **Q** powracamy do głównego menu.

2.4.4. Rozpoczęcie instalacji standardowej

Instalacja standardowa zalecana jest dla wszystkich zaczynających swą przygodę z FreeBSD, bądź w ogóle z systemem UNIX. Klawiszami kursora wybieramy Standard i wciskamy **Enter**.

Rysunek 2-10. Rozpoczęcie instalacji standardowej

```
FreeBSD/3.0.0 8.1-RELEASE - sysinstall Main Menu
Welcome to the FreeBSD installation and configuration tool. Please
select one of the options below by using the arrow keys or typing the
first character of the option name you're interested in. Invoke an
option with [SPACE] or [ENTER]. To exit, use [TAB] to move to Exit.

Usage: Quick start - How to use this menu system
Standard Begin a standard installation (recommended)
Express Begin a quick installation (for experts)
Custom Begin a custom installation (for experts)
Configure Do post-install configuration of FreeBSD
Doc Installation instructions, README, etc.
Keymap Select keyboard type
Options View/Set various installation options
Fixit Repair mode with CDRUM/DVD/floppy or start shell
Upgrade Upgrade an existing system
Load Config... Load default install configuration
Index Glossary of functions

[ Select ] X Exit Install
```

2.5. Przydział miejsca na dysku

Zaczynamy od przydzielenia FreeBSD przestrzeni dyskowej, oraz oznaczenia tej przestrzeni w taki sposób, by **sysinstall** mógł ją przygotować. Do tego potrzebna nam będzie wiedza na temat sposobu, w jaki FreeBSD znajduje informacje zapisane na dysku.

2.5.1. Kolejność dysków w BIOS-ie

Przed instalacją i konfiguracją FreeBSD powinniśmy zapoznać się z pewnym ważnym zagadnieniem, szczególnie istotnym dla posiadaczy dwóch lub więcej twardych dysków.

W komputerze typu PC wyposażonym w zależny od BIOS-u system operacyjny, jak na przykład MS-DOS lub Microsoft Windows, BIOS może zmienić rzeczywistą kolejność dysków, a system operacyjny tę zmianę zaakceptuje. Dzięki temu system może zostać uruchomiony z dysku innego niż tzw. „primary master”. Jest to szczególnie wygodne dla tych użytkowników, którzy za najprostszą i najtańszą metodę tworzenia kopii zapasowej uważają kupno identycznego drugiego twardego dysku i kopiowanie zawartości pierwszego dysku przy użyciu **Ghost®** lub **XCOPY**. W przypadku uszkodzenia pierwszego dysku, ataku wirusa lub awarii systemu operacyjnego, dane mogą być z łatwością odzyskane poprzez zamianę logicznej kolejności dysków w BIOS-ie. To tak, jakby zamienić przewody dysków, ale bez konieczności otwierania obudowy.

Droższe maszyny wyposażone w kontrolery SCSI mają często rozszerzenia BIOS-u pozwalające zamieniać kolejność dysków SCSI na podobnej zasadzie, obsługując do siedmiu dysków.

Użytkowników przyzwyczajonych do korzystania z tego typu rozwiązań może spotkać niespodzianka, gdy we FreeBSD rezultaty odbiegają od oczekiwań. FreeBSD nie korzysta z BIOS-u, jak również nie zna „logicznej kolejności dysków BIOS-u”. W efekcie może to prowadzić do kłopotliwych sytuacji, szczególnie wtedy, gdy dyski są identyczne pod względem geometrii, oraz zawierają takie same dane.

Planując używanie FreeBSD, powinniśmy ustawić w BIOS-ie rzeczywistą kolejność dysków przed instalacją systemu, i tę kolejność pozostawić. Jeśli chcemy koniecznie zamienić dyski, to możemy to zrobić sprzętowo, otwierając obudowę i zamieniając odpowiednie zworki i przewody.

Fragment z Archiwum Wyjątkowych Przygód Bolka i Lolka:

Bolek ma przygotować dla Lolka komputer z FreeBSD. Bolek montuje jeden dysk SCSI jako urządzenie SCSI zero, i instaluje na nim FreeBSD.

Lolek zaczyna korzystać z systemu, ale po kilku dniach zauważa, że dysk SCSI zgłasza liczne błędy, więc zawiadamia o tym Bolka.

Po kolejnych kilku dniach Bolek postanawia rozwiązać problem, więc bierze ze „składzika” taki sam dysk SCSI. Kontrola powierzchni dysku wykazuje, że dysk działa prawidłowo, więc Bolek podłącza go jako czwarte urządzenie SCSI i wykonuje kopię dysku zerowego na dysk czwarty. Ponieważ dysk jest podłączony i działa jak należy, Bolek stwierdza, że można zacząć go używać, więc wykorzystując możliwości BIOS-u SCSI zmienia kolejność dysków w taki sposób, by system uruchamiany był z czwartego urządzenia SCSI. FreeBSD uruchamia się i działa jak należy.

Lolek korzysta z systemu przez jakiś czas, następnie wspólnie z Bolkiem postanawiają spróbować czegoś nowego — zainstalować nowszą wersję FreeBSD. Bolek wymontowuje dysk SCSI zero, ponieważ działał kiepsko, i zastępuje go kolejnym identycznym dyskiem ze „składzika”. Bolek instaluje nową wersję FreeBSD na nowym dysku SCSI korzystając z czarodziejskich dyskietek instalacyjnych Lolka. Instalacja przebiega prawidłowo.

Lolek używa nowej wersji FreeBSD przez parę dni i stwierdza, że można zacząć korzystać z niej w pracy. Wcześniej jednak trzeba będzie skopiować wszystkie dane ze starej wersji. Lolek podłącza więc czwarty dysk SCSI (najświeższą kopię starej wersji FreeBSD). Lolek stwierdza jednak z niepokojem, że na dysku nie ma śladu po jego cennych danych.

Gdzie się one podziały?

Gdy Bolek sporządził kopię dysku zerowego na dysku czwartym, dysk czwarty stał się „klonem”. Zmieniając kolejność dysków w BIOS-ie SCSI aby móc uruchamiać system z dysku czwartego, Bolek sam siebie wprowadzał w błąd. FreeBSD wciąż działało na dysku zerowym. Zmiana w BIOS-ie powoduje, że część kodu uruchamiającego FreeBSD jest rzeczywiście ładowana z dysku wskazanego w BIOS-ie, lecz kiedy pałeczkę przejmują sterowniki jądra FreeBSD, kolejność dysków BIOS-u przestaje obowiązywać, a FreeBSD przechodzi z powrotem na rzeczywistą kolejność. W opowiadanej historyjce system nadal działał na dysku zerowym, i tam właśnie znajdowały się cenne dane Lolka, a nie na dysku czwartym. Choć wydawało się, że system działa na dysku czwartym, było to tylko złudzenie.

Z przyjemnością oznajmiamy, iż ani jeden bajt cennych danych nie zginął ani nie został w inny sposób skrzywdzony podczas naszych badań nad opisanym zjawiskiem. Stary dysk SCSI zero został odnaleziony i cenne dane wróciły do Lolka (Bolek z kolei przekonał się, że niczego nie można być pewnym).

W opowieści udział wzięły dyski SCSI, jednakże w przypadku dysków IDE sytuacja wyglądałaby tak samo.

2.5.2. Tworzenie segmentów za pomocą programu FDisk

Notatka: Dokonywane tutaj zmiany nie zostaną zapisane na dysku. Jeżeli będziemy podejrzewać, że coś zrobiliśmy źle, możemy wybrać w menu wyjście z programu **sysinstall** i spróbować jeszcze raz od początku, bądź wcisnąć **U** by skorzystać z opcji Undo (cofnij). W ostateczności, jeżeli całkiem stracimy orientację, możemy po prostu wyłączyć komputer.

Po wybraniu standardowej instalacji w **sysinstall** zostanie wyświetlony następujący komunikat:

Message

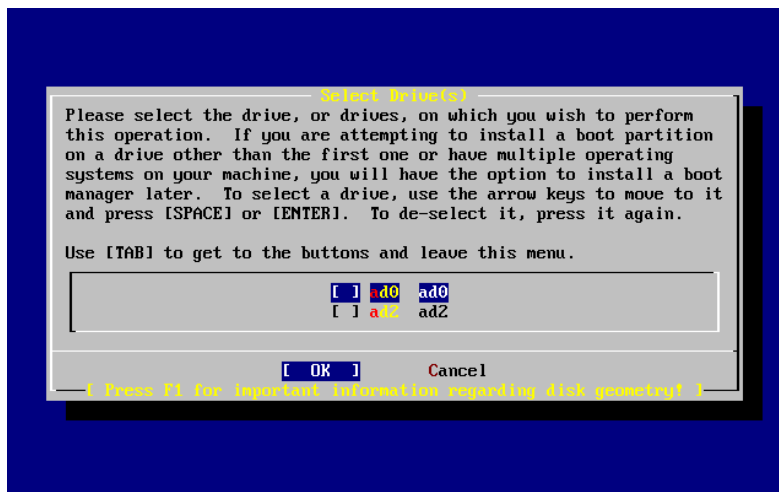
In the next menu, you will need to set up a DOS-style ("fdisk") partitioning scheme for your hard disk. If you simply wish to devote all disk space to FreeBSD (overwriting anything else that might be on the disk(s) selected) then use the (A)ll command to select the default partitioning scheme followed by a (Q)uit. If you wish to allocate only free space to FreeBSD, move to a partition marked "unused" and use the (C)reate command.

[OK]

[Press enter or space]

Zgodnie z poleceniem naciskamy **Enter**. Zobaczymy teraz listę twardej dysków znalezionych przez jądro podczas rozpoznawania urządzeń. Rysunek 2-11 przedstawia przykład komputera z dwoma dyskami IDE, o nazwach ad0 i ad2.

Rysunek 2-11. Wybór dysku FDisk-a



Można się zastanawiać, dlaczego na liście brakuje ad1. Co spowodowało, że został pominięty?

Przyjmijmy przykładowo, że mamy dwa dyski IDE, jeden jako master na pierwszym kontrolerze IDE, drugi jako master na drugim kontrolerze IDE. Gdyby we FreeBSD zostały one ponumerowane w takiej kolejności, w jakiej zostały wykryte, czyli ad0 i ad1, wszystko działałoby jak należy.

Gdybyśmy jednak zainstalowali potem jeszcze jeden dysk, jako slave na pierwszym kontrolerze IDE, to ten właśnie dysk zostałby nowym ad1, a wcześniejszy ad1 zmieniłby się w ad2. Ponieważ systemy plików odnajdywane są według nazw urządzeń (np. ad1s1a), mogłoby się nagle okazać, że niektóre systemy plików nie działają poprawnie. Aby to poprawić, musielibyśmy zmienić konfigurację systemu.

Aby zapobiec takim sytuacjom, jądro FreeBSD może być skonfigurowane tak, by przydzielać dyskom IDE numery zgodne z ich rzeczywistym umiejscowieniem, niezależnie od kolejności wykrywania. Tym sposobem dysk podłączony jako master na drugim kontrolerze IDE *zawsze* będzie mieć nazwę ad2, nawet w sytuacji, gdy ad0 i ad1 nie są w ogóle obecne.

Jądro FreeBSD domyślnie skonfigurowane jest właśnie w ten sposób, dlatego też na ekranie mamy `ad0` i `ad2`. Komputer, z którego ten rysunek pochodzi, miał dwa dyski IDE podłączone jako master do obu kontrolerów IDE, nie miał natomiast dysków podłączonych jako slave.

Wybieramy dysk, na którym chcemy zainstalować FreeBSD i wybieramy [OK]. Zostanie uruchomiony **FDisk**, pokazując na ekranie obraz podobny do Rysunek 2-12.

Ekran **FDisk**-a podzielony jest na trzy części.

Część pierwsza, obejmująca pierwsze dwie linie ekranu, zawiera informacje o wybranym dysku, w tym jego oznaczenie we FreeBSD, geometrię oraz całkowity rozmiar dysku.

Druga część pokazuje informacje o istniejących na dysku segmentach: gdzie się one zaczynają oraz kończą, jaki jest ich rozmiar, jaka nazwa została im nadana przez FreeBSD ich opis oraz typ. Na rysunku przykładowym widać dwa niewielkie nieużywane segmenty, obecne ze względu na stosowany w architekturze PC podział dysku. Prócz tego widać duży segment FAT, który prawie na pewno jest dyskiem `C`: w MS-DOS / Windows, oraz segment rozszerzony, zawierający być może dyski MS-DOS / Windows oznaczone kolejnymi literami.

W trzeciej części znajduje się lista dostępnych w **FDisk**-u poleceń.

Rysunek 2-12. Układ partycji w **FDisk**-u przed zmianami

```

Disk name:      ad0                      FDISK Partition Editor
DISK Geometry:  16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset      Size(ST)      End      Name  PType      Desc  Subtype  Flags
-----
0           63           62      -      6      unused    0
63         4193217       4193279  ad0s1  2      fat      14      >
4193280      1008       4194287  -      6      unused    0      >
4194288     12319776      16514063 ad0s2  4      extended  15      >

The following commands are supported (in upper or lower case):

A = Use Entire Disk      G = set Drive Geometry  C = Create Slice      F = 'DD' mode
D = Delete Slice         Z = Toggle Size Units   S = Set Bootable      I = Wizard m.
T = Change Type          U = Undo All Changes    Q = Finish

Use F1 or ? to get more help, arrow keys to select.

```

Dalej postępować będziemy w zależności od tego, jak chcemy podzielić nasz dysk na segmenty.

Jeżeli chcemy, by FreeBSD zajęło cały dysk (co wiąże się z usunięciem z niego wszelkich innych danych, gdy potwierdzimy to w **sysinstall** na późniejszym etapie instalacji), naciskamy **A**, co odpowiada opcji **Use Entire Disk** (wykorzystaj cały dysk). Istniejące segmenty zostaną usunięte, a w ich miejsce pojawi się mały obszar opisany jako **unused** (nieużywany; znów jest to następstwem pecetowego układu dysku), oraz duży segment przeznaczony dla FreeBSD. Jeżeli decydujemy się na tę opcję, powinniśmy w następnej kolejności wskazać nowoutworzony segment FreeBSD przy użyciu klawiszy kursora i wcisnąć **S**, by umożliwić ładowanie systemu z tego segmentu. Ekran będzie wyglądał podobnie do przedstawionego na Rysunek 2-13. Zwróćmy uwagę na literę **A** w kolumnie **Flags**, oznacza ona, że segment jest *aktywny* i będzie z niego ładowany system.

Jeśli chcemy usunąć istniejący segment by zwolnić miejsce dla FreeBSD, wskazujemy segment korzystając z klawiszy kursora i naciskamy **D**. Następnie możemy nacisnąć **C** i w odpowiedzi na pytanie o rozmiar segmentu, który chcemy utworzyć, wpisać odpowiednią wartość i wcisnąć **Enter**. Wartość domyślna stanowi największy możliwy rozmiar segmentu, czyli np. wolną przestrzeń na dysku bądź całą pojemność dysku twardego.

Wolne miejsce dla FreeBSD mogliśmy także przygotować wcześniej (na przykład przy użyciu programu **PartitionMagic**), w takim wypadku po prostu wciskamy **C** by utworzyć nowy segment. W tym przypadku również zostaniemy zapytani o rozmiar segmentu, który zamierzamy stworzyć.

Rysunek 2-13. Partycja w FDisk-u obejmująca cały dysk

```

Disk name:      ad0                      FDISK Partition Editor
DISK Geometry: 16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset      Size(ST)      End      Name  PType      Desc  Subtype  Flags
-----
0           63           62      -      6      unused     0
63      16514001      16514063  ad0s1  3      freebsd    165     CA

The following commands are supported (in upper or lower case):

A = Use Entire Disk      G = set Drive Geometry  C = Create Slice  F = 'DD' mode
D = Delete Slice         Z = Toggle Size Units   S = Set Bootable  I = Wizard m.
T = Change Type          U = Undo All Changes    Q = Finish

Use F1 or ? to get more help, arrow keys to select.

```

Na koniec naciskamy **Q**. Dokonane zmiany zostaną zapamiętane przez **sysinstall**, ale nie będą jeszcze zapisane na dysku.

2.5.3. Instalacja programu ładującego

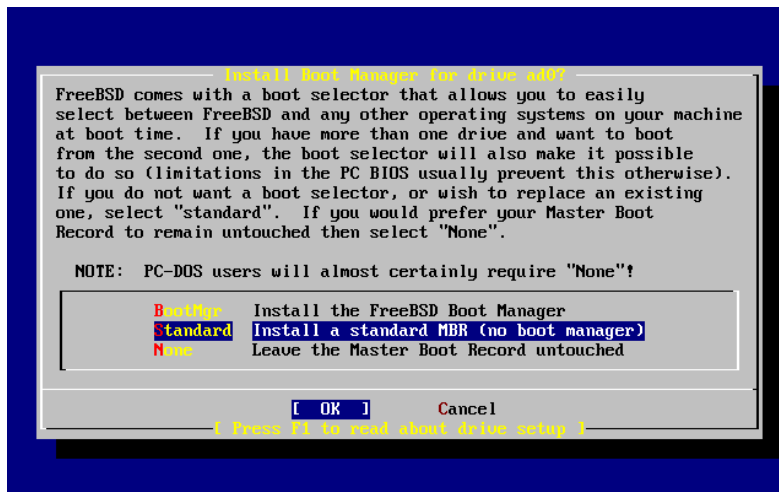
W kolejnym kroku instalacji będziemy mieć możliwość zainstalowania programu ładującego (ang. boot manager). Mówiąc ogólnie, powinniśmy instalować program ładujący FreeBSD jeżeli:

- Mamy dwa lub więcej dysków, a FreeBSD instalujemy na dysku innym niż pierwszy.
- Instalujemy FreeBSD obok innego systemu operacyjnego na tym samym dysku, i chcemy mieć możliwość wybrania systemu operacyjnego podczas uruchamiania komputera.

Jeśli FreeBSD będzie jedynym systemem operacyjnym na danym komputerze i zostanie zainstalowany na pierwszym dysku twardym, wówczas wystarczy wykorzystać **Standardowy** program ładujący. Natomiast jeśli wykorzystujemy już inny program potrafiący uruchomić FreeBSD powinniśmy wybrać opcję **None** (żaden).

Dokonany wybór potwierdzamy naciskając **Enter**.

Rysunek 2-14. Wybór programu ładującego w sysinstall



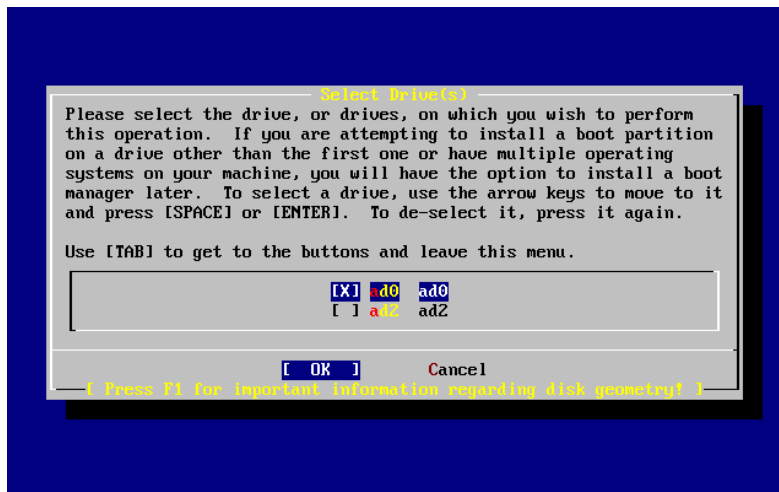
Ekran pomocy, wyświetlany po naciśnięciu **F1**, opisuje problemy z jakimi można się spotkać, gdy planuje się mieć kilka systemów operacyjnych na jednym dysku.

2.5.4. Tworzenie segmentów na innym dysku

Jeżeli mamy więcej dysków, po wyborze programu ładującego ponownie ukaże się ekran wyboru dysku. Chcąc zainstalować FreeBSD na kilku dyskach, wybieramy tutaj kolejny dysk i ponownie korzystając z programu **FDisk** tworzymy na nim segmenty.

WAŻNE: Jeśli instalujemy FreeBSD na innym dysku niż pierwszy, wówczas program ładujący FreeBSD musi zostać zainstalowany na obydwu dyskach.

Rysunek 2-15. Zakończenie wyboru dysku



Klawisz **Tab** przełącza pomiędzy ostatnio wybranym dyskiem oraz przyciskami [OK], i [Cancel].

Wciskamy **Tab** jeden raz, by wybrać [OK], następnie naciskamy **Enter** aby przejść do kolejnego etapu instalacji.

2.5.5. Tworzenie partycji z wykorzystaniem Disklabel

W nowoutworzonych segmentach musimy stworzyć kilka partycji. Pamiętajmy, że każda partycja oznaczona jest literą od a do h, a partycje b, c i d rządzą się specjalnymi zasadami, których należy przestrzegać.

Niektóre aplikacje mogą skorzystać na stosowaniu określonych schematów podziału na partycje, szczególnie, gdy partycje rozłożone są na kilku dyskach. Na razie jednak, ponieważ jest to nasza pierwsza instalacja FreeBSD, nie powinniśmy zbytnio przejmować się podziałem dysku na partycje. Ważniejszym jest, byśmy zainstalowali FreeBSD i zaczęli się uczyć, jak go używać. Kiedy już nabierzemy pewnej wprawy, możemy zainstalować system ponownie i zmienić sposób podziału na partycje.

Poniższy schemat przedstawia cztery partycje — jedną dla przestrzeni wymiany, oraz trzy dla systemów plików.

Tabela 2-2. Układ partycji pierwszego dysku

Partycja	System plików	Rozmiar	Opis
a	/	100 MB	Będzie to główny system plików. Wszystkie inne systemy plików będą zamontowane gdzieś wewnątrz niego. 100 MB jest dość rozsądnym rozmiarem dla tego celu. Nie będzie tu przechowywane zbyt wiele danych, zwykle po instalacji FreeBSD umieszcza tu około 40 MB danych. Pozostałe miejsce jest dla danych tymczasowych, oraz służy jako zapas, gdyby kolejne wersje FreeBSD potrzebowały więcej miejsca w /.

Partycja	System plików	Rozmiar	Opis
b	brak	2-3 x RAM	<p>Partycja ta służy jako przestrzeń wymiany. Wybór jej odpowiedniego rozmiaru nie jest sprawą banalną. Możemy przyjąć, że przestrzeń wymiany powinna być dwu- lub trzykrotnie większa niż ilość pamięci fizycznej (RAM). Prócz tego powinniśmy mieć co najmniej 64 MB przestrzeni wymiany, więc jeżeli nasz komputer ma mniej niż 32 MB pamięci, ustawmy rozmiar przestrzeni wymiany na 64 MB.</p> <p>Jeśli dysponujemy kilkoma dyskami, możemy na każdym z nich umieścić przestrzeń wymiany. FreeBSD będzie w procesie wymiany wykorzystywać każdy z dysków, dzięki czemu wymiana będzie się odbywać szybciej. W takim przypadku przyjmujemy całkowity rozmiar potrzebnej przestrzeni wymiany (np. 128 MB) i dzielimy go przez liczbę posiadanych dysków (np. dwa dyski), otrzymując w wyniku rozmiar przestrzeni wymiany dla jednego dysku. W naszym przykładzie będzie to 64 MB na każdy dysk.</p>
e	/var	50 MB	<p>W katalogu /var przechowywane są pliki o zmiennych rozmiarach; pliki dzienników systemowych i inne pliki administracyjne. Podczas codziennej pracy FreeBSD na wielu z tych plików dokonywane są częste operacje odczytu lub zapisu. Dzięki umieszczeniu ich w oddzielnym systemie plików FreeBSD może dokonać optymalizacji dostępu do nich, nie wywierając jednocześnie wpływu na inne pliki, do których dostęp przebiega inaczej.</p>
f	/usr	Reszta dysku	Inne pliki będą zwykle przechowywane w katalogu /usr i jego podkatalogach.

Jeżeli instalujemy FreeBSD na dwóch lub więcej dyskach, musimy utworzyć partycje także w innych przygotowanych segmentach. Najłatwiej jest po prostu przygotować na każdym z kolejnych dysków dwie partycje, jedną na przestrzeń wymiany, drugą na system plików.

Tabela 2-3. Układ partycji dla kolejnych dysków

Partycja	System plików	Rozmiar	Opis
b	brak	Patrz: opis	<p>Jak już powiedzieliśmy, przestrzeń wymiany możemy dzielić między kilka dysków. Mimo, iż mamy do dyspozycji partycję a, zgodnie z obowiązującą konwencją przestrzeń wymiany powinna znajdować się na partycji b.</p>

Partycja	System plików	Rozmiar	Opis
e	/dyskn	Reszta dysku	Pozostała część dysku zajmowana jest przez jedną dużą partycję. Mogłaby to z powodzeniem być partycja a, zamiast e. Przyjęto jednak, że partycja a zarezerwowana jest dla głównego systemu plików (/). Nie ma przymusu stosowania tej zasady, jednak sysinstall jej przestrzega, dobrze więc jest ją stosować dla zachowania porządku podczas instalacji. System plików możemy zamontować w dowolnym miejscu, w przykładzie zaproponowano /dyskn, gdzie <i>n</i> jest kolejnym numerem każdego dysku. Można jednak wybrać inne nazewnictwo według uznania..

Po podjęciu decyzji jak ma wyglądać układ partycji, pora wprowadzić go w życie używając **sysinstall**. Na ekranie ukaże się następujący komunikat:

```

                                Message
Now, you need to create BSD partitions inside of the fdisk
partition(s) just created. If you have a reasonable amount of disk
space (200MB or more) and don't have any special requirements, simply
use the (A)uto command to allocate space automatically. If you have
more specific needs or just don't care for the layout chosen by
(A)uto, press F1 for more information on manual layout.

                                [ OK ]
                                [ Press enter or space ]

```

Naciskamy **Enter** by przejść do edytora partycji FreeBSD, zwanego **Disklabel**.

Rysunek 2-16 przedstawia ekran zaraz po uruchomieniu **Disklabel**. Jest on podzielony na trzy części.

W kilku pierwszych wierszach widoczna jest nazwa wybranego aktualnie dysku, oraz nazwa segmentu, w którym tworzymy partycje (**Disklabel** używa tutaj nazwy *Partition name*, czyli nazwa partycji, a nie nazwa segmentu). Jest tu również zawarta informacja o rozmiarze wolnej przestrzeni wewnątrz segmentu, czyli przestrzeni nie przydzielonej jeszcze partycjom.

Środek ekranu zajmuje lista utworzonych partycji, wraz z nazwami przechowywanych na nich systemów plików, ich rozmiarami oraz pewnymi opcjami związanymi z tworzeniem systemu plików.

W dolnej części przedstawiona jest lista dostępnych w **Disklabel** poleceń.

Rysunek 2-16. Edytor Disklabel

```

FreeBSD Disklabel Editor

Disk: ad0      Partition name: ad0s1      Free: 16514001 blocks (8063MB)

Part      Mount      Size Newfs      Part      Mount      Size Newfs
-----
-----

The following commands are valid here (upper or lower case):
C = Create      D = Delete      M = Mount pt.
N = Newfs Opts  Q = Finish      S = Toggle SoftUpdates  Z = Custom Newfs
T = Toggle Newfs U = Undo      A = Auto Defaults      R = Delete+Merge

Use F1 or ? to get more help, arrow keys to select.

```

Disklabel potrafi automatycznie utworzyć partycje i nadać im domyślne rozmiary. Wypróbujmy tę możliwość naciskając **A**. Na ekranie ukaże się obraz podobny do Rysunek 2-17. Ustawienia automatyczne mogą być właściwe lub nie, w zależności od rozmiaru dysku. Nie ma to jednak większego znaczenia, ponieważ nie trzeba ich koniecznie akceptować.

Notatka: Katalog `/tmp` jest domyślnie umieszczany na własnej partycji, zamiast być częścią partycji `/`. Dzięki temu można uniknąć zapelnienia partycji `/` plikami tymczasowymi.

Rysunek 2-17. Edytor disklabel z automatycznymi ustawieniami

```

FreeBSD Disklabel Editor

Disk: ad0      Partition name: ad0s1      Free: 0 blocks (0MB)

Part      Mount      Size Newfs      Part      Mount      Size Newfs
-----
-----
ad0s1a    /           422MB UFS2      Y
ad0s1b    swap        321MB SWAP
ad0s1d    /var        710MB UFS2+S  Y
ad0s1e    /tmp        377MB UFS2+S  Y
ad0s1f    /usr        6232MB UFS2+S  Y

The following commands are valid here (upper or lower case):
C = Create      D = Delete      M = Mount pt.
N = Newfs Opts  Q = Finish      S = Toggle SoftUpdates  Z = Custom Newfs
T = Toggle Newfs U = Undo      A = Auto Defaults      R = Delete+Merge

Use F1 or ? to get more help, arrow keys to select.

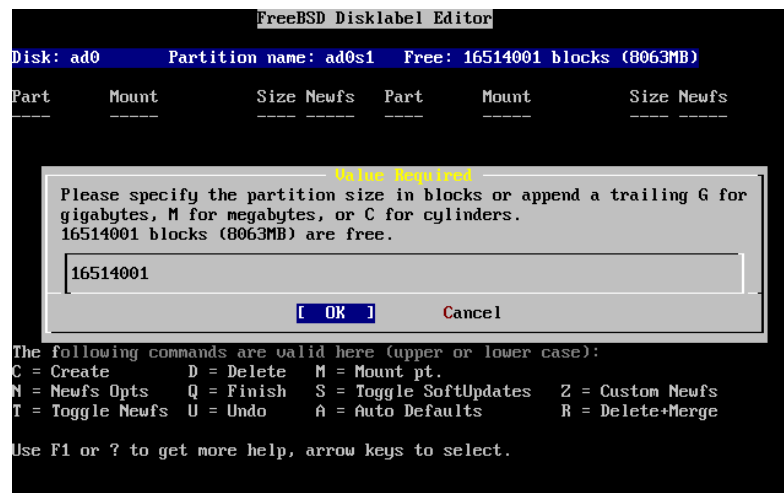
```

By usunąć zaproponowane partycje i zastąpić je utworzonymi własnoręcznie, wybieramy klawiszami kursora pierwszą partycję i naciskamy **D**. Tak samo postępujemy z pozostałymi partycjami.

Teraz, aby stworzyć pierwszą partycję (a, zamontowaną jako /), wybieramy informacje o dysku w górnej części ekranu i wciskamy **C**. Pojawi się okienko z pytaniem o rozmiar nowej partycji (Rysunek 2-18). Wybrany rozmiar podać możemy w blokach, albo w wygodniejszej formie w postaci liczby megabajtów, gigabajtów lub cylindrów, odpowiednio z przyrostkiem M, G lub C.

Notatka: Począwszy od FreeBSD 5.X użytkownicy mogą: wybrać system plików UFS2 (domyślny system we FreeBSD 5.1 i późniejszych) wykorzystując opcję *Custom Newfs* (**Z**), tworzyć partycje za pomocą *Auto Defaults* i modyfikować przy pomocy *Custom Newfs* bądź dodać opcję `-o 2` podczas normalnego procesu tworzenia partycji. Wykorzystując opcję *Custom Newfs* musimy pamiętać by dodać flagę `-U` (*SoftUpdates*)!

Rysunek 2-18. Wolne miejsce dla głównej partycji



Wybierając domyślnie zaproponowany rozmiar utworzymy partycję obejmującą pozostałe miejsce w segmencie. Jeżeli zamierzamy stworzyć partycje o takich rozmiarach, jak wcześniej opisywaliśmy, wówczas kasujemy zaproponowaną wartość klawiszem **Backspace**, i wpisujemy 64M, Rysunek 2-19. Następnie wybieramy **[OK]**.

Rysunek 2-19. Zmiana rozmiaru głównej partycji



Po wybraniu rozmiaru partycji pojawi się pytanie, czy partycja zawierać będzie system plików, czy przestrzeń wymiany. Okienko z tym pytaniem pokazane jest na Rysunek 2-20. Pierwsza partycja zawierać będzie system plików, wybieramy więc FS i naciskamy **Enter**.

Rysunek 2-20. Wybór typu głównej partycji



Ponieważ na partycji znajdować się będzie system plików, **Disklabel** musi wiedzieć, gdzie będzie on zamontowany. Rysunek 2-21 przedstawia okienko z prośbą o podanie tej informacji. Główny system plików montowany jest jako /, wpisujemy więc / i wciskamy **Enter**.

Rysunek 2-21. Wybór miejsca montowania głównego systemu plików



Na ekranie pojawi się informacja o nowo utworzonej partycji. Powinniśmy teraz powtórzyć całą procedurę dla kolejnych partycji. Tworząc partycję wymiany nie będziemy pytani o miejsce jej zamontowania, ponieważ partycje wymiany nie są montowane. Gdy będziemy tworzyć ostatnią partycję, `/usr`, możemy przyjąć proponowany rozmiar domyślny, aby przeznaczyć na tę partycję resztę segmentu.

Ostatecznie ekran edytora Disklabel będzie wyglądać podobnie do Rysunek 2-22, choć wybrane przez nas wartości mogą być inne. By zakończyć pracę z Disklabel, wciskamy **Q**.

Rysunek 2-22. Edytor Disklabel



2.6. Wybór składników instalacji

2.6.1. Wybór zestawu komponentów

Decyzja o tym, jaki zestaw komponentów zainstalujemy, zależy w dużej mierze od planowanych zastosowań systemu i ilości wolnego miejsca na dysku. Dostępne warianty pozwalają zarówno na instalację najmniejszej konfiguracji, jak i na instalację wszystkiego. Początkujący użytkownicy systemów UNIX i FreeBSD powinni wybrać jeden z przygotowanych wariantów. Dla bardziej doświadczonych użytkowników istnieje możliwość ułożenia własnego zestawu komponentów.

Więcej informacji o zestawach komponentów i ich zawartości możemy uzyskać naciskając **F1**. Po przejrzaniu tych informacji naciskamy **Enter**, aby powrócić do menu wyboru komponentów.

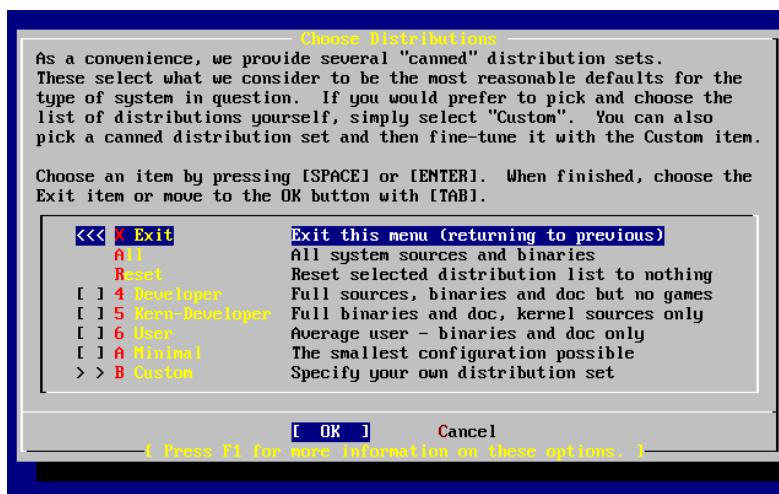
Jeśli planujemy korzystać z graficznego interfejsu użytkownika powinniśmy wybrać jeden z zestawów o nazwie rozpoczynającej się literą x. Po instalacji zajmiemy się konfigurowaniem serwera graficznego i wyborem menedżera okien. Szczegółowe informacje na ten temat zawiera rozdział Rozdział 5.

To, która wersja systemu X11 jest domyślnie instalowana, zależy od instalowanej wersji FreeBSD. Wydania wcześniejsze od 5.3 domyślnie instalują **XFree86 4.X**. Natomiast FreeBSD 5.3 i późniejsze instalują **Xorg**.

Jeżeli planujemy samodzielne kompilowanie jądra, powinniśmy wybrać wariant zawierający kod źródłowy. Rozdział 8 zawiera informacje, dlaczego powinno się budować niestandardowe jądro i jak to zrobić.

Oczywiście najbardziej wszechstronny jest system zawierający wszystkie komponenty. Jeśli mamy wystarczająco dużo miejsca na dysku, wybieramy klawiszami kursora **All**, Rysunek 2-23, i naciskamy **Enter**. Jeżeli jednak miejsca na dysku mogłoby nie wystarczyć, wybierzmy wariant najlepiej odpowiadający obecnym potrzebom. Kolejne komponenty mogą być dodawane po zainstalowaniu systemu.

Rysunek 2-23. Wybór komponentów



2.6.2. Instalacja kolekcji portów

Po wyborze komponentów będziemy mieć możliwość zainstalowania kolekcji portów FreeBSD. Kolekcja portów umożliwia łatwe i wygodne instalowanie oprogramowania. Nie zawiera ona kodów źródłowych programów. W skład

kolekcji portów wchodzi pliki umożliwiające automatyczne pobieranie programów, oraz ich kompilowanie i instalowanie. Rozdział 4 opisuje sposób korzystanie z kolekcji portów.

Program instalacyjny nie sprawdza, czy mamy odpowiednio dużo wolnego miejsca na dysku. Kolekcję portów powinniśmy instalować tylko pod warunkiem, że miejsca faktycznie wystarczy. We FreeBSD 9.1 kolekcja zajmuje około 500 MB.

```

User Confirmation Requested
Would you like to install the FreeBSD ports collection?

```

```

This will give you ready access to over 23,000 ported software packages,
at a cost of around 500 MB of disk space when "clean" and possibly much
more than that if a lot of the distribution tarballs are loaded
(unless you have the extra CDs from a FreeBSD CD/DVD distribution
available and can mount it on /cdrom, in which case this is far less
of a problem).

```

```

The Ports Collection is a very valuable resource and well worth having
on your /usr partition, so it is advisable to say Yes to this option.

```

```

For more information on the Ports Collection & the latest ports,
visit:

```

```

http://www.FreeBSD.org/ports

```

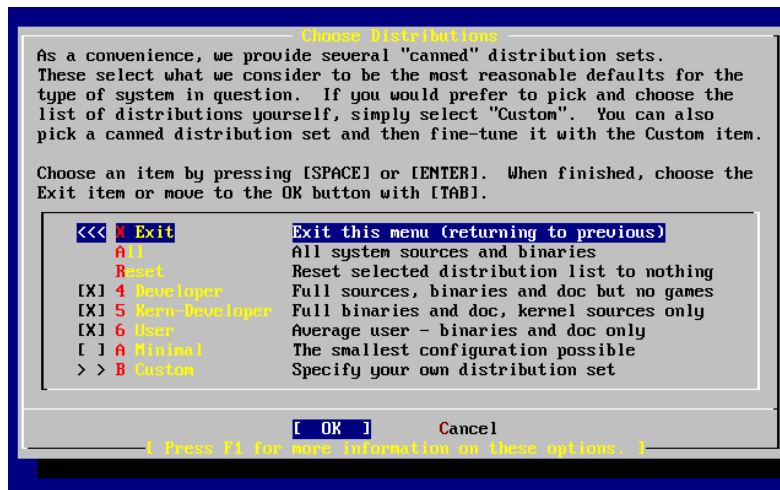
```

[ Yes ]      No

```

Klawiszami kursora wybieramy [Yes], aby zainstalować kolekcję portów, lub [No], by z niej zrezygnować. Wybór zatwierdzamy klawiszem **Enter**. Ponownie pojawi się menu wyboru komponentów.

Rysunek 2-24. Zatwierdzenie wybranych komponentów



Jeżeli odpowiadają nam wybrane komponenty, przy pomocy klawiszy kursora wybieramy Exit, zaznaczamy [OK] i naciskamy **Enter**, przechodząc do kolejnego etapu instalacji.

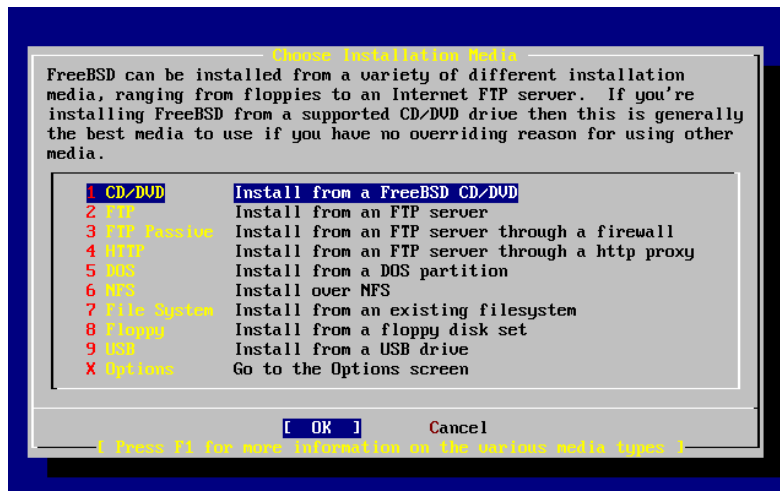
2.7. Wybór nośnika instalacji

W przypadku, gdy instalujemy z płyty CD bądź DVD, klawiszami kursora wybieramy pozycję **Install from a FreeBSD CD/DVD** (instalacja z CD/DVD). Upewniwszy się, że zaznaczone jest [OK], naciskamy **Enter** przechodząc do następnego etapu instalacji.

Jeżeli stosujemy inną metodę instalacji, wybieramy odpowiednią pozycję i postępujemy zgodnie ze wskazówkami.

Klawiszem **F1** możemy włączyć pomoc. Do menu wyboru nośnika powracamy naciskając **Enter**.

Rysunek 2-25. Wybór nośnika instalacji



Tryby instalacji przez FTP: Można wybrać jeden z trzech trybów instalacji przez FTP: aktywne FTP, pasywne FTP lub pośrednio przez HTTP proxy.

Aktywne FTP: Install from an FTP server

Wybór tego wariantu spowoduje, że przesyłanie danych przez FTP odbywać się będzie w trybie „aktywnym”. Nie zadziała to w przypadku transmisji przez zaporę ogniową, ale będzie współpracować ze starszymi serwerami FTP nie obsługującymi trybu pasywnego. Jeśli połączenie pasywne (wybierane domyślnie) nie zadziała, spróbujmy aktywnego!

Pasywne FTP: Install from an FTP server through a firewall

Opcja ta informuje **sysinstall**, że przesyłanie danych przez FTP odbywać się będzie w trybie „pasywnym”. Pozwoli to na połączenie poprzez zaporę ogniową, która nie zezwala na połączenia z zewnątrz z portami o przypadkowych numerach.

FTP przez proxy HTTP: Install from an FTP server through a http proxy

Ten wariant instruuje **sysinstall** do wykorzystania protokołu HTTP (podobnie jak przeglądarka stron WWW) do połączenia się z serwerem proxy pośredniczącym w transmisji przez FTP. Serwer pośredniczący przetwarza żądania i przesyła je do serwera FTP. Dzięki temu możliwe jest połączenie poprzez zaporę

ogniową nie zezwalającą na żadne połączenia FTP, oferującą jednak HTTP proxy. W takiej sytuacji, poza adresem serwera FTP, będziemy musieli podać także adres serwera proxy.

Korzystając z pośredniczącego serwera FTP proxy, zwykle podajemy nazwę serwera docelowego jako część nazwy użytkownika, po znaku „@”. Serwer proxy „udaje” wówczas serwer docelowy. Załóżmy, dla przykładu, że chcemy zainstalować system z `ftp.FreeBSD.org`, za pośrednictwem serwera proxy FTP `foo.example.com`, nasłuchującego na porcie 1024.

W takiej sytuacji przechodzimy do menu opcji, jako nazwę użytkownika FTP wpisujemy `ftp@ftp.FreeBSD.org`, a jako hasło podajemy nasz adres email. Jako nośnik instalacji wybieramy FTP (lub pasywne FTP, jeżeli umożliwia to serwer proxy), a jako URL wpisujemy `ftp://foo.example.com:1234/pub/FreeBSD`.

Ze względu na to, że `/pub/FreeBSD` z `ftp.FreeBSD.org` jest udostępnione na serwerze proxy `foo.example.com`, możemy właśnie z tego serwera dokonać instalacji (ponieważ zajmie się on pobraniem odpowiednich plików z `ftp.FreeBSD.org`).

2.8. Przystąpienie do instalacji

Możemy teraz rozpocząć właściwą instalację, a zarazem mamy ostatnią szansę na rezygnację z instalacji bez zmiany zawartości dysku twardego.

```

User Confirmation Requested
Last Chance! Are you SURE you want to continue the installation?

If you're running this on a disk with data you wish to save then WE
STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!

We can take no responsibility for lost disk contents!

[ Yes ]      No

```

Wybieramy `[Yes]` i wciskamy **Enter**, by rozpocząć instalację.

Czas trwania instalacji zależy od wybranych komponentów, używanego nośnika instalacji oraz prędkości komputera. Szereg komunikatów informować będzie o przebiegu procesu instalacji.

Po zakończeniu instalacji wyświetlony zostanie następujący komunikat:

```

Message

Congratulations! You now have FreeBSD installed on your system.

We will now move on to the final configuration questions.
For any option you do not wish to configure, simply select No.

If you wish to re-enter this utility after the system is up, you may
do so by typing: /stand/sysinstall .

```

```

[ OK ]

[ Press enter to continue ]

```

Po naciśnięciu klawisza **Enter** zajmiemy się przygotowaniem wstępnej konfiguracji systemu.

Jeśli wybierzemy [No] i naciśniemy **Enter** instalacja zostanie przerwana, bez dokonywania jakichkolwiek zmian. Pojawi się komunikat o treści:

```

                        Message
Installation complete with some errors.  You may wish to scroll
through the debugging messages on VTY1 with the scroll-lock feature.
You can also choose "No" at the next prompt and go back into the
installation menus to retry whichever operations have failed.

                        [ OK ]

```

Powyższy komunikat pojawia się, ponieważ nic nie zostało zainstalowane. Naciskając **Enter** możemy powrócić do głównego menu i opuścić program instalacyjny.

2.9. Po instalacji

Po pomyślnie zakończonej instalacji zajmiemy się wstępną konfiguracją systemu. Wszelkich zmian w ustawieniach możemy dokonać przed uruchomieniem nowo zainstalowanego systemu FreeBSD lub też po zakończeniu instalacji, korzystając z `sysinstall` (we FreeBSD starszych niż 5.2 `/stand/sysinstall`) i jego opcji **Configure**.

2.9.1. Konfiguracja urządzeń sieciowych

Jeśli wcześniej skonfigurowaliśmy PPP na potrzeby instalacji przez FTP, konfiguracja urządzeń sieciowych zostanie pominięta. Będziemy mogli zająć się nią później.

Szczegółowe informacje na temat sieci lokalnych (LAN) oraz konfiguracji FreeBSD w roli bramy lub routera znaleźć można w rozdziale **Zaawansowana konfiguracja sieciowa**.

```

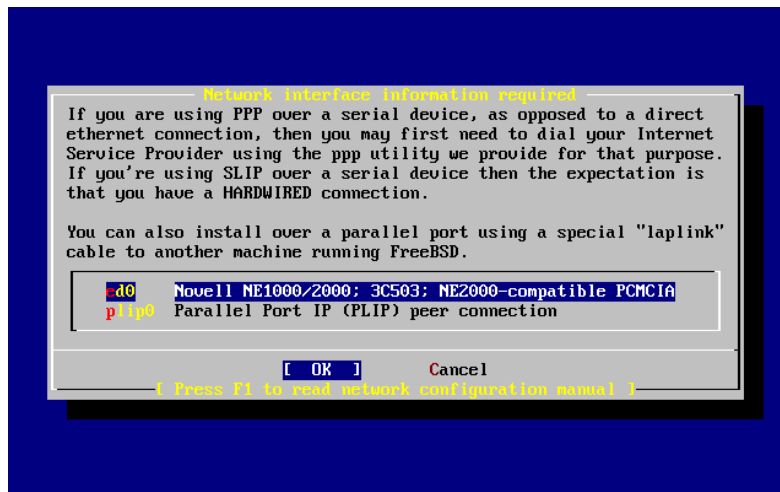
                        User Confirmation Requested
Would you like to configure any Ethernet or SLIP/PPP network devices?

                        [ Yes ]   No

```

Jeśli chcemy skonfigurować urządzenie sieciowe, wybieramy [Yes] i wciskamy **Enter**. W przeciwnym wypadku wybieramy [No].

Rysunek 2-26. Wybór karty Ethernet



Klawiszami kursora wybieramy interfejs, który będziemy konfigurować i wciskamy **Enter**.

```

User Confirmation Requested
Do you want to try IPv6 configuration of the interface?

Yes    [ No ]

```

Dla przykładu, w sieci lokalnej w zupełności wystarcza obecny protokół Internetu (IPv4), wybieramy więc klawiszami kursora [**No**] i naciskamy **Enter**.

Jeśli chcemy wypróbować nowy protokół Internetu (IPv6), wybieramy [**Yes**] i naciskamy **Enter**. Przez chwilę będzie się odbywać poszukiwanie serwerów RA.

```

User Confirmation Requested
Do you want to try DHCP configuration of the interface?

Yes    [ No ]

```

Jeżeli nie wykorzystujemy DHCP (Dynamic Host Configuration Protocol), wybieramy klawiszami kursora [**No**] i wciskamy **Enter**.

Wybranie [**Yes**] spowoduje uruchomienie **dhclient** i jeśli wszystko przebiegnie prawidłowo, konfiguracja sieci zostanie rozpoznana automatycznie. Sekcja 25.5 zawiera szczegółowe informacje na ten temat.

Przedstawiony poniżej ekran konfiguracji sieci (Network Configuration) przedstawia konfigurację karty sieciowej komputera, który będzie służył jako brama w sieci lokalnej.

Rysunek 2-27. Konfiguracja interfejsu ed0

Klawiszem **Tab** wybieramy poszczególne pola, w których wpisujemy odpowiednie informacje:

Host (stacja)

Pełna nazwa stacji, w powyższym przykładzie `k6-2.example.com`.

Domain (domena)

Nazwa domeny, do której należy stacja, w przykładzie jest to `example.com`.

IPv4 Gateway (brama IPv4)

Adres IP stacji przekazującej pakiety do odbiorców spoza sieci lokalnej. Musi być podany, jeśli komputer jest węzłem w sieci. Jeżeli komputer pełni rolę bramy do Internetu w sieci lokalnej, pole to należy *pozostawić puste*.

Name server (serwer nazw)

Adres IP lokalnego serwera DNS. W przykładowej sieci lokalnej nie ma serwera DNS, wpisany więc został adres serwera DNS dostawcy Internetu (`208.163.10.2`).

IPv4 address (adres IPv4)

W przykładzie temu interfejsowi przypisano adres `192.168.0.1`.

Netmask (maska podsieci)

W sieci lokalnej użyty został dla przykładu blok adresów klasy C (`192.168.0.0 - 192.168.0.255`). Maską podsieci jest maska sieci klasy C (`255.255.255.0`).

Extra options to ifconfig (dodatkowe opcje dla ifconfig)

Tu wpisywane są dodatkowe opcje dla `ifconfig` charakterystyczne dla interfejsu. W pokazanym przykładzie nie było takowych opcji.

Gdy konfiguracja będzie gotowa, klawiszem **Tab** wybieramy [OK] i naciskamy **Enter**.

User Confirmation Requested

```
Would you like to Bring Up the ed0 interface right now?
```

```
[ Yes ]    No
```

Jeśli wybierzemy [Yes] i wciśniemy **Enter**, komputer zostanie aktywowany do pracy w sieci.

2.9.2. Konfiguracja bramy

```
User Confirmation Requested
```

```
Do you want this machine to function as a network gateway?
```

```
[ Yes ]    No
```

Jeśli komputer będzie w sieci lokalnej pełnić rolę bramy, czyli będzie przekazywać pakiety pomiędzy innymi komputerami, wybieramy opcję [Yes] i naciskamy **Enter**. Jeżeli natomiast komputer będzie węzłem w sieci, wybieramy [No] i również wciskamy **Enter**.

2.9.3. Konfiguracja usług internetowych

```
User Confirmation Requested
```

```
Do you want to configure inetd and the network services that it provides?
```

```
Yes      [ No ]
```

Wybranie [No] spowoduje, że wiele usług (jak np. **telnetd**) będą wyłączone. Oznacza to, że zdalni użytkownicy nie będą mogli połączyć się z naszym komputerem za pomocą **telnetu**. Użytkownicy lokalni będą natomiast mogli łączyć się z odległymi komputerami korzystając z **telnetu**.

Usługi możemy włączyć po zainstalowaniu systemu, aby to zrobić, modyfikujemy plik `/etc/inetd.conf` za pomocą edytora tekstu. Więcej informacji znaleźć można w Sekcji 25.2.1. Sekcja 25.2.1.

Jeśli wolelibyśmy skonfigurować usługi internetowe podczas instalacji, wybieramy [Yes]. Zostaniemy poproszeni o dodatkowe potwierdzenie:

```
User Confirmation Requested
```

```
The Internet Super Server (inetd) allows a number of simple Internet
services to be enabled, including finger, ftp and telnetd.  Enabling
these services may increase risk of security problems by increasing
the exposure of your system.
```

```
With this in mind, do you wish to enable inetd?
```

```
[ Yes ]    No
```

Wybieramy [Yes], by przejść dalej.

```
User Confirmation Requested
```

```
inetd(8) relies on its configuration file, /etc/inetd.conf, to determine
which of its Internet services will be available.  The default FreeBSD
inetd.conf(5) leaves all services disabled by default, so they must be
specifically enabled in the configuration file before they will
```

function, even once inetd(8) is enabled. Note that services for IPv6 must be separately enabled from IPv4 services.

Select [Yes] now to invoke an editor on /etc/inetd.conf, or [No] to use the current settings.

[Yes] No

Wybranie [Yes] pozwoli na włączanie poszczególnych usług poprzez usunięcie znaku # na początku właściwego wiersza.

Rysunek 2-28. Modyfikacja inetd.conf

```

^_ (escape) menu  ^y search prompt  ^k delete line    ^p prev li       ^g prev page
^o ascii code     ^x search         ^l undelete line  ^n next li       ^v next page
^u end of file    ^a begin of line  ^w delete word    ^b back 1 char
^t top of text    ^e end of line    ^r restore word   ^f forward 1 char
^c command        ^d delete char    ^j undelete char  ^z next word

=====line 1 col 0 lines from top 1 =====
# $FreeBSD: src/etc/inetd.conf,v 1.73.10.2.4.1 2010/06/14 02:09:06 kensmith Exp
#
# Internet server configuration database
#
# Define *both* IPv4 and IPv6 entries for dual-stack support.
# To disable a service, comment it out by prefixing the line with '#'.
# To enable a service, remove the '#' at the beginning of the line.
#
#ftp      stream  tcp        nowait  root    /usr/libexec/ftpd      ftpd -l
#ftp      stream  tcp6       nowait  root    /usr/libexec/ftpd      ftpd -l
#ssh      stream  tcp        nowait  root    /usr/sbin/sshd         sshd -i -4
#ssh      stream  tcp6       nowait  root    /usr/sbin/sshd         sshd -i -6
#telnet   stream  tcp        nowait  root    /usr/libexec/telnetd   telnetd
#telnet   stream  tcp6       nowait  root    /usr/libexec/telnetd   telnetd
#shell    stream  tcp        nowait  root    /usr/libexec/rshd      rshd
#shell    stream  tcp6       nowait  root    /usr/libexec/rshd      rshd
#login    stream  tcp        nowait  root    /usr/libexec/rlogind   rlogind
#login    stream  tcp6       nowait  root    /usr/libexec/rlogind   rlogind
file "/etc/inetd.conf", 118 lines

```

Gdy włączymy wybrane usługi, naciskamy **Esc** by przejść do menu, w którym będziemy mogli zakończyć modyfikowanie pliku i zapisać zmiany.

2.9.4. Anonimowe FTP

User Confirmation Requested

Do you want to have anonymous FTP access to this machine?

Yes [No]

2.9.4.1. Wyłączenie anonimowego FTP

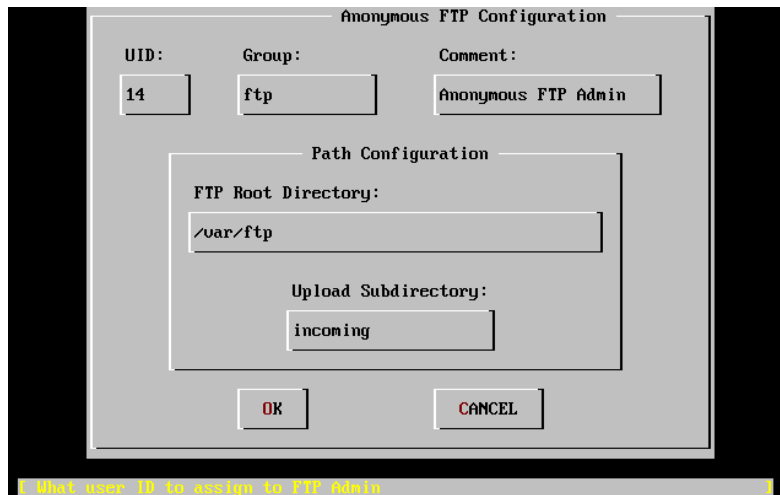
Wybranie zaznaczonego domyślnie [No] pozwoli na dostęp do komputera poprzez FTP tylko tym użytkownikom, którzy mają własne konta chronione hasłem.

2.9.4.2. Włączenie anonimowego FTP

Włączenie anonimowego FTP oznacza, że każdy będzie mógł uzyskać dostęp do komputera. Zanim się na to zdecydujemy, powinniśmy być świadomi niebezpieczeństwa, które się z tym wiąże. Rozdział 14 zawiera więcej informacji na temat bezpieczeństwa.

Aby włączyć anonimowe FTP, klawiszami kursora wybieramy [Yes] i naciskamy **Enter**. Ekran będzie wyglądać jak na poniższym rysunku (lub podobnie):

Rysunek 2-29. Domyślne ustawienia anonimowego FTP



Możemy nacisnąć **F1**, by uzyskać pomoc:

This screen allows you to configure the anonymous FTP user.

The following configuration values are editable:

UID: The user ID you wish to assign to the anonymous FTP user.
All files uploaded will be owned by this ID.

Group: Which group you wish the anonymous FTP user to be in.

Comment: String describing this user in /etc/passwd

FTP Root Directory:

Where files available for anonymous FTP will be kept.

Upload subdirectory:

Where files uploaded by anonymous FTP users will go.

Główny katalog ftp jest domyślnie umieszczany w /var. Jeżeli nie mamy tam wystarczająco dużo miejsca dla przewidywanych potrzeb FTP, możemy wybrać w zamian katalog /usr, jako główny katalog FTP (FTP Root Directory) wpisując /usr/ftp.

Po wybraniu odpowiadających nam ustawień naciskamy **Enter**.

```

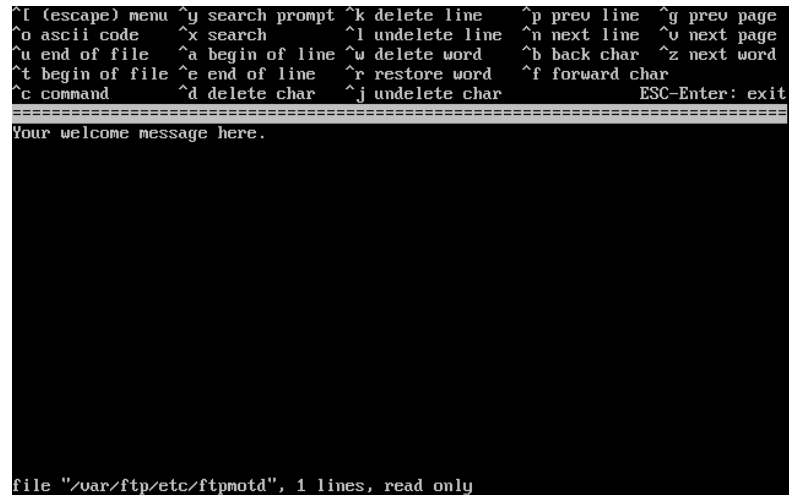
User Confirmation Requested
Create a welcome message file for anonymous FTP users?

[ Yes ]      No

```

Jeżeli wybierzemy [Yes] i wciśniemy **Enter**, automatycznie zostanie uruchomiony edytor, w którym będziemy mogli napisać komunikat powitalny dla użytkowników anonimowego FTP.

Rysunek 2-30. Edycja komunikatu powitalnego FTP



Używany tutaj edytor tekstu jest `ee`. Postępując zgodnie z przedstawionymi na ekranie wskazówkami możemy wprowadzić treść komunikatu, lub też możemy zrobić to później, korzystając z dowolnego edytora. W tym celu warto jest zapisać nazwę i lokalizację pliku pokazywaną na dole ekranu.

Gdy naciśniemy **Esc** pokazane zostanie menu z domyślnie zaznaczoną opcją `a`) `leave editor`. (opuszczenie edytora). Wybieramy ją naciskając **Enter**. Ponowne naciśnięcie **Enter** spowoduje zapisanie zmian jeśli jakichś dokonaliśmy.

2.9.5. Konfiguracja sieciowych usług plikowych

Sieciowe usługi plikowe (Network File Services - NFS) pozwalają na współdzielony dostęp do plików przez sieć. Komputer możemy skonfigurować jako serwer, klient, lub oba naraz. Więcej informacji na ten temat można znaleźć w Sekcja 25.3.

2.9.5.1. Serwer NFS

```

User Confirmation Requested
Do you want to configure this machine as an NFS server?

Yes      [ No ]

```

Jeśli nie zamierzamy korzystać z serwera NFS, wybieramy [No] i wciskamy **Enter**.

W przeciwnym wypadku, gdy wybierzemy [Yes], zostanie pokazany komunikat o konieczności stworzenia pliku exports.

Message

Operating as an NFS server means that you must first configure an /etc/exports file to indicate which hosts are allowed certain kinds of access to your local filesystems.

Press [Enter] now to invoke an editor on /etc/exports

[OK]

Naciskamy **Enter**. Zostanie uruchomiony edytor tekstu, w którym będziemy mogli przygotować plik exports.

Rysunek 2-31. Edycja pliku exports

```

^I (escape) menu  ^y search prompt  ^k delete line    ^p prev li      ^g prev page
^o ascii code    ^x search         ^l undelete line  ^n next li      ^u next page
^u end of file    ^a begin of line  ^w delete word    ^b back 1 char
^t begin of file  ^e end of line    ^r restore word   ^f forward 1 char
^c command        ^d delete char    ^j undelete char  ^z next word
L: 1 C: 1 =====
#The following examples export /usr to 3 machines named after ducks,
#/usr/src and /usr/ports read-only to machines named after trouble makers
#/home and all directories under it to machines named after dead rock stars
#and, /a to a network of privileged machines allowed to write on it as root.
#/usr          huey louie dewie
#/usr/src /usr/obj -ro  calvin hobbes
#/home -alldirs      janice jimmy frank
#/a      -maproot=0 -network 10.0.1.0 -mask 255.255.248.0
#
# You should replace these lines with your actual exported filesystems.
# Note that BSD's export syntax is 'host-centric' vs. Sun's 'FS-centric' one.

file "/etc/exports", 12 lines

```

Zgodnie ze wskazówkami dopisujemy udostępniane systemy plików. Możemy także zrobić to później, korzystając z preferowanego przez nas edytora tekstu. W tym celu warto zapisać sobie pokazywaną na dole ekranu nazwę i lokalizację pliku.

Gdy naciśniemy **Esc**, pokazane zostanie menu z domyślnie zaznaczoną opcją a) leave editor (opuszczenie edytora). Wybieramy ją naciskając **Enter**.

2.9.5.2. Klient NFS

Instalacja klienta NFS pozwoli naszemu komputerowi łączyć się z serwerami NFS.

User Confirmation Requested

Do you want to configure this machine as an NFS client?

Yes [No]

Wybieramy klawiszami kursora [Yes] lub [No] zależnie od podjętej decyzji, po czym naciskamy **Enter**.

2.9.6. Profil zabezpieczeń

„Profil zabezpieczeń” to zestaw opcji konfiguracyjnych, mający zapewnić określony poziom bezpieczeństwa poprzez włączenie i wyłączenie pewnych programów i ustawień. Im surowszy profil zabezpieczeń, tym mniej programów będzie domyślnie uruchamianych. Odpowiada to jednej z podstawowych zasad bezpieczeństwa: należy wyłączać wszystko, co nie musi być włączone.

Pamiętajmy, że profil zabezpieczeń to tylko domyślne ustawienia. Poszczególne programy można włączać i wyłączać już po zainstalowaniu FreeBSD, poprzez modyfikację lub dodanie odpowiednich wpisów w pliku `/etc/rc.conf`. Dalsze informacje na ten temat znaleźć można w dokumentacji systemowej `rc.conf(5)`.

Poniższa tabela pokazuje, jaki jest efekt stosowania każdego z profili zabezpieczeń. Kolumny odpowiadają profilom, które można wybrać, natomiast w kolejnych wierszach wymienione są poszczególne programy lub funkcje włączone lub wyłączone w danym profilu.

Tabela 2-4. Dostępne profile zabezpieczeń

	Extreme	Medium
sendmail(8)	NIE	TAK
sshd(8)	NIE	TAK
portmap(8)	NIE	MOŻE ^a
serwer NFS	NIE	TAK
securelevel(8)	TAK ^b	NIE

Przypisy: a. Portmapper jest włączony, jeśli na wcześniejszym etapie instalacji komputer został skonfigurowany jako klient lub serwer.

```

User Confirmation Requested
Do you want to select a default security profile for this host (select
No for "medium" security)?

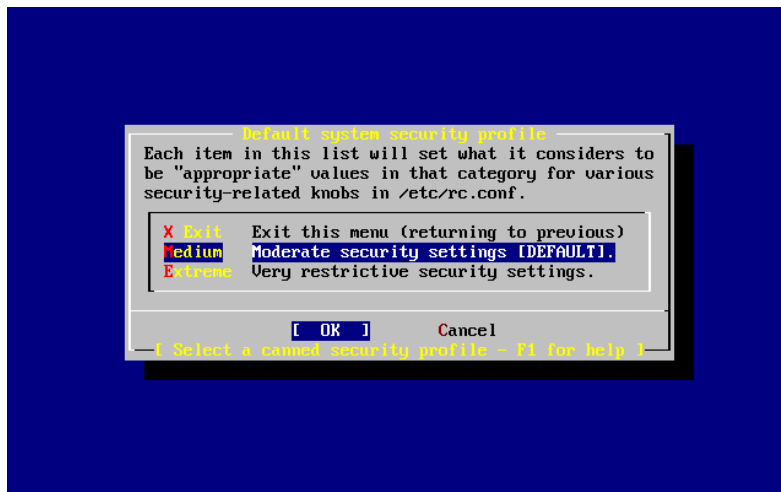
[ Yes ]      No

```

Jeżeli wybierzemy [No] i naciśniemy **Enter**, zostanie ustawiony średni profil zabezpieczeń.

Chcąc wybrać inny profil zabezpieczeń, wybieramy [Yes] i wciskamy **Enter**.

Rysunek 2-32. Opcje profilu zabezpieczeń



Aby uzyskać pomoc, wciskamy **F1**. Naciskając **Enter** wracamy do menu.

Klawiszami kursora wybieramy **Medium**, chyba, że jesteśmy pewni, że będziemy potrzebować innego poziomu bezpieczeństwa. Wskazujemy następnie **[OK]** i wciskamy **Enter**.

Zostanie wyświetlony komunikat potwierdzający wybór profilu zabezpieczeń.

Message

Moderate security settings have been selected.

Sendmail and SSHd have been enabled, securelevels are disabled, and NFS server setting have been left intact. PLEASE NIETE that this still does not save you from having to properly secure your system in other ways or exercise due diligence in your administration, this simply picks a standard set of out-of-box defaults to start with.

To change any of these settings later, edit /etc/rc.conf

[OK]

Message

Extreme security settings have been selected.

Sendmail, SSHd, and NFS services have been disabled, and securelevels have been enabled. PLEASE NIETE that this still does not save you from having to properly secure your system in other ways or exercise due diligence in your administration, this simply picks a more secure set of out-of-box defaults to start with.

To change any of these settings later, edit /etc/rc.conf

[OK]

Naciskamy **Enter**, aby przejść do kolejnego etapu konfiguracji.

Ostrzeżenie Profil zabezpieczeń nie jest cudownym lekarstwem! Nawet, jeśli wybraliśmy najbardziej bezpieczny profil, musimy na bieżąco interesować się sprawami bezpieczeństwa systemu, czytając poświęcone im listy dyskusyjne (Sekcja C.1), stosując dobre hasła i przestrzegając ogólnych zasad bezpieczeństwa. Profil jest tylko wygodnym sposobem na przygotowanie podstawowych zabezpieczeń.

2.9.7. Ustawienia konsoli systemowej

Kilka opcji służy do konfiguracji konsoli systemowej.

```

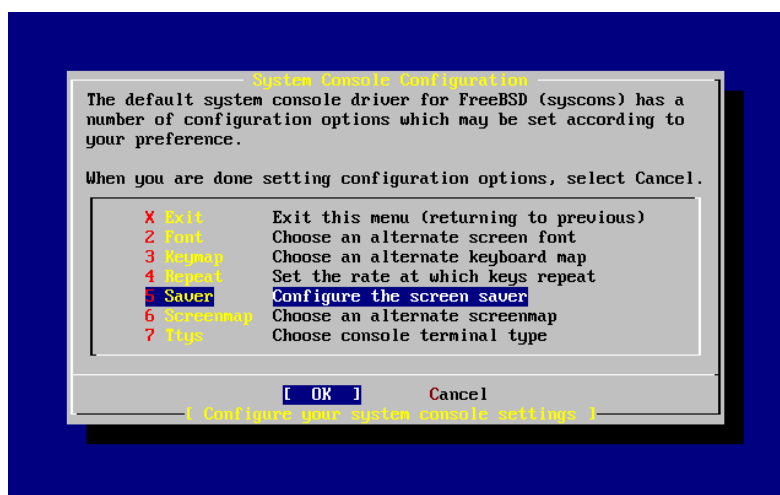
User Confirmation Requested
Would you like to customize your system console settings?

[ Yes ] No

```

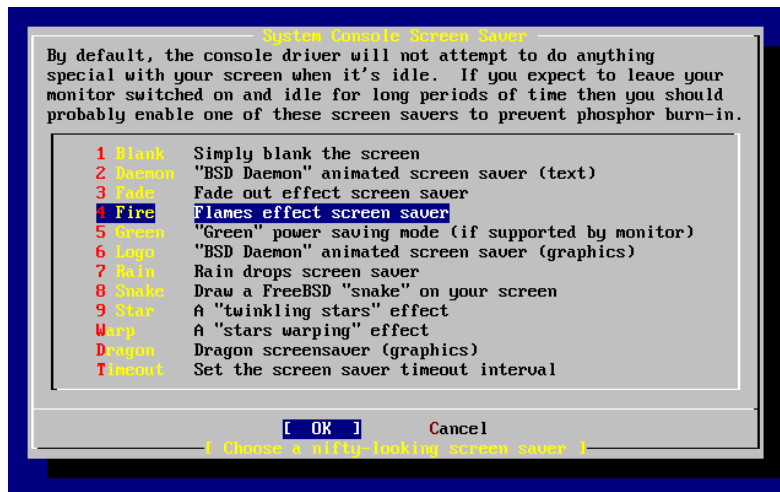
Aby zobaczyć i zmienić ustawienia, wybieramy [Yes] i wciskamy **Enter**.

Rysunek 2-33. Opcje konfiguracji konsoli systemowej



Często stosowaną opcją jest wygaszacz ekranu (screen saver). Klawiszami kursora wybieramy **Saver** i naciskamy **Enter**.

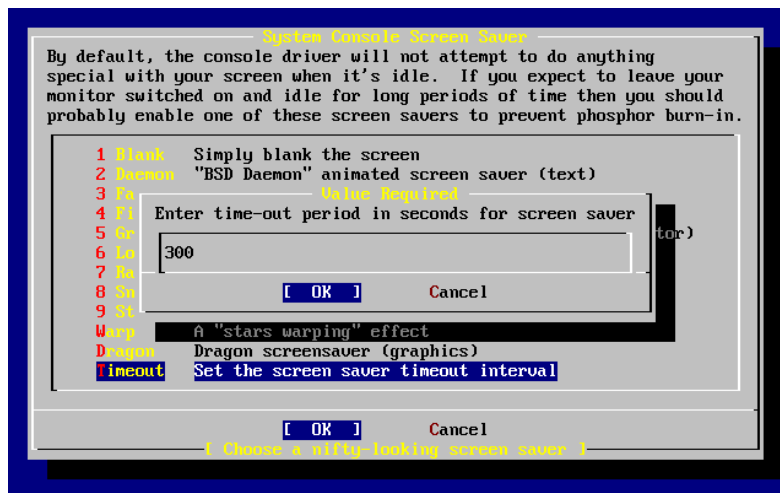
Rysunek 2-34. Opcje wygaszacza ekranu



Za pomocą klawiszy kursora wybieramy odpowiadający nam wygaszacz i wciskamy **Enter**. Ponownie pojawi się menu konfiguracji konsoli systemowej.

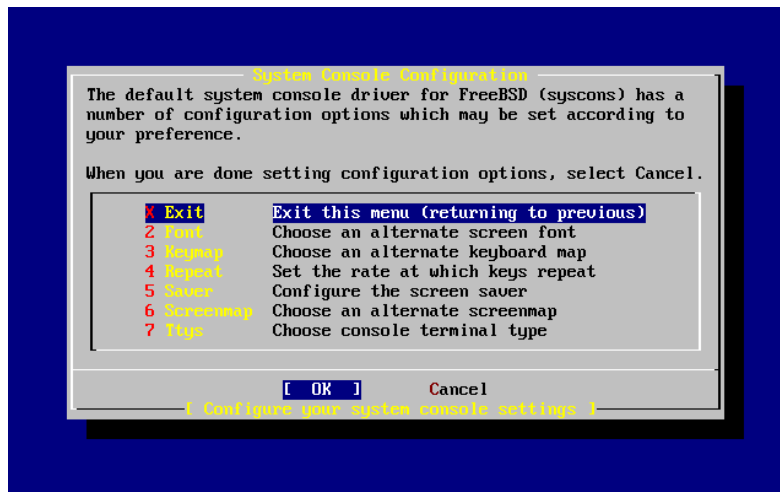
Przyjmowany domyślnie przedział czasu wynosi 300 sekund. Aby go zmienić, ponownie wybieramy Saver. W menu opcji wygaszacza ekranu klawiszami kursora wybieramy Timeout i naciskamy **Enter**. Pojawi się okienko:

Rysunek 2-35. Limit czasu wygaszacza ekranu



Wartość możemy zmienić, po czym wybieramy [OK] i wciskamy **Enter**, by wrócić do menu konfiguracji konsoli.

Rysunek 2-36. Zakończenie konfiguracji konsoli



Wybieramy **Exit** i naciskamy **Enter**, przechodząc do kolejnego etapu konfiguracji.

2.9.8. Ustawienia strefy czasowej

Dzięki ustawieniu strefy czasowej komputer będzie mógł automatycznie ustawiać zegar w przypadku zmiany czasu, jak również będzie prawidłowo wykonywać inne czynności związane ze strefą czasową.

W przykładzie mamy do czynienia z komputerem znajdującym się we wschodniej strefie czasowej Stanów Zjednoczonych. Rzeczywiste ustawienia będą zależać od naszego położenia geograficznego.

```
User Confirmation Requested
Would you like to set this machine's time zone now?

[ Yes ]   No
```

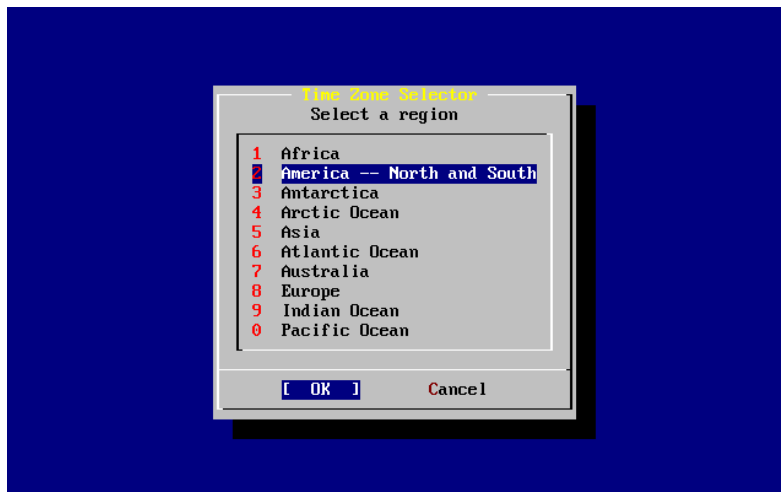
By ustawić strefę czasową, wybieramy **[Yes]** i naciskamy **Enter**.

```
User Confirmation Requested
Is this machine's CMOS clock set to UTC? If it is set to local time
or you don't know, please choose NIE here!

Yes    [ No ]
```

Wybieramy **[Yes]** lub **[No]**, w zależności od ustawienia zegara komputera, następnie wciskamy **Enter**.

Rysunek 2-37. Wybór regionu geograficznego



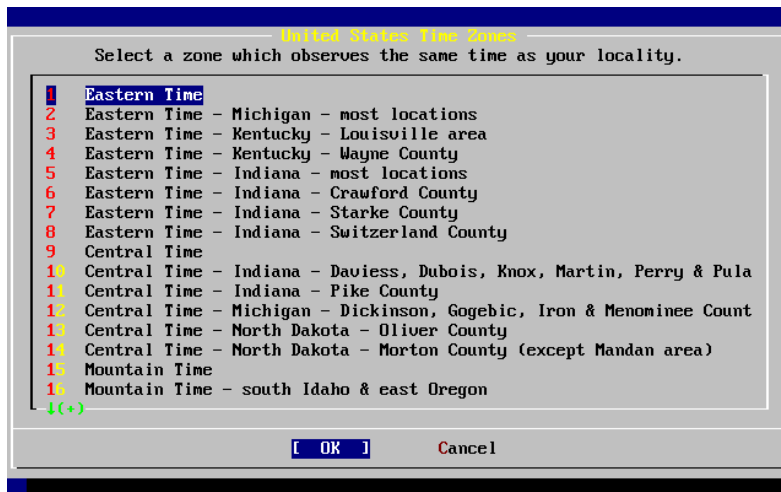
Klawiszami kursora wybieramy odpowiedni region, po czym naciskamy **Enter**.

Rysunek 2-38. Wybór kraju



Przy użyciu klawiszy kursora wybieramy odpowiedni kraj i naciskamy **Enter**.

Rysunek 2-39. Wybór strefy czasowej



Klawiszami kursora wybieramy właściwą strefę czasową i wciskamy **Enter**.

```
Confirmation
Does the abbreviation 'EDT' look reasonable?

[ Yes ]   No
```

Zostaniemy zapytani, czy skrót nazwy strefy czasowej jest prawidłowy. Jeśli tak, naciskamy **Enter** i przechodzimy do kolejnego etapu konfiguracji.

2.9.9. Kompatybilność z Linuksem

```
User Confirmation Requested
Would you like to enable Linux binary compatibility?

[ Yes ]   No
```

Wybranie [Yes] i naciśnięcie **Enter** pozwoli uruchamiać programy linuxowe we FreeBSD. Program instalacyjny dołączy pakiety obsługujące kompatybilność z Linuksem.

Jeśli instalujemy system przez FTP, komputer będzie potrzebować łączności z Internetem. Może się zdarzyć, że na serwerze ftp będzie brakowało pewnych składników, na przykład obsługujących kompatybilność z Linuksem. Można je jednak zainstalować później.

2.9.10. Ustawienia myszki

Posługując się 3-przyciskową myszką będziemy mogli wycinać i wklejać tekst na konsoli i w uruchamianych programach. Jeśli nasza myszka ma dwa przyciski, po instalacji zajrzyjmy do dokumentacji systemowej moused(8), gdzie opisana została emulacja trzech przycisków. W naszym przykładzie konfigurujemy myszkę nie podłączoną przez USB (np. przez złącze PS/2 lub port COM)::

```

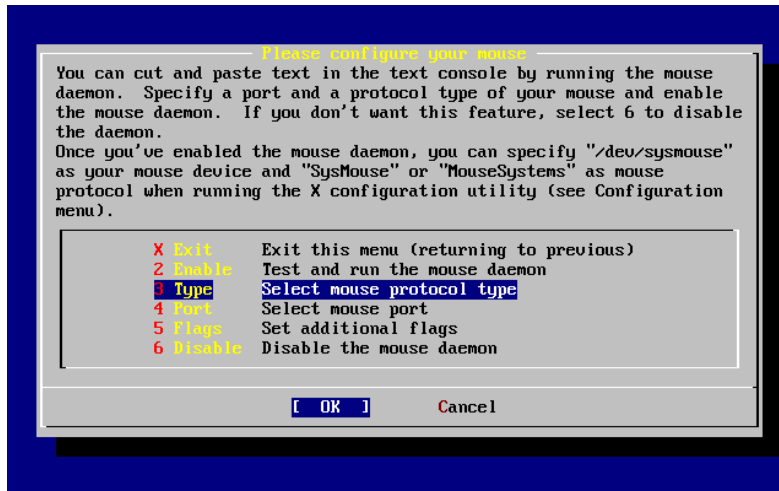
User Confirmation Requested
Does this system have a non-USB mouse attached to it?

[ Yes ]      No

```

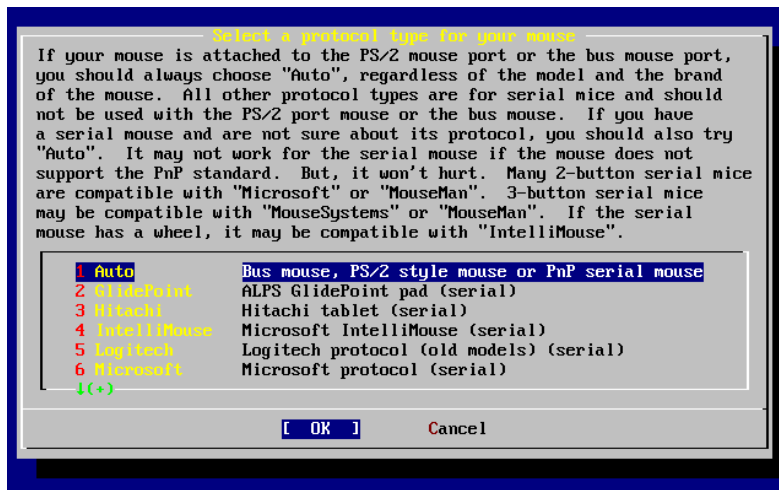
Wybieramy [No], jeśli myszka podłączona jest przez USB, lub [Yes] w przeciwnym wypadku i naciskamy **Enter**.

Rysunek 2-40. Opcja wyboru protokołu myszki



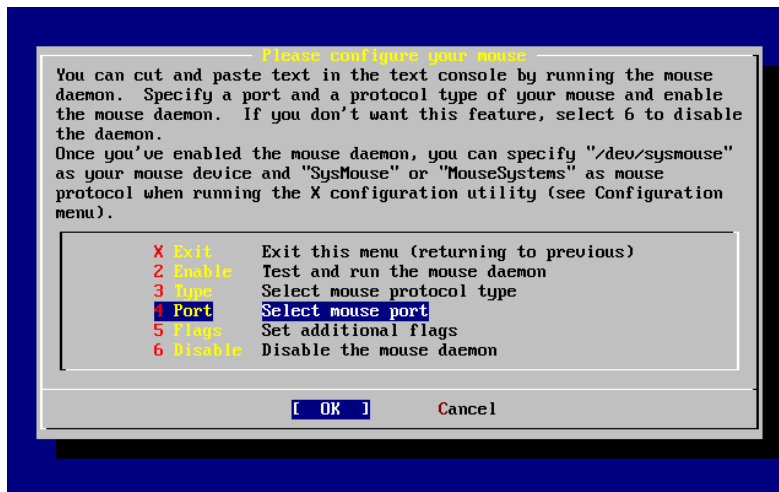
Klawiszami kursora wskazujemy **Type** i naciskamy **Enter**.

Rysunek 2-41. Wybór protokołu myszki



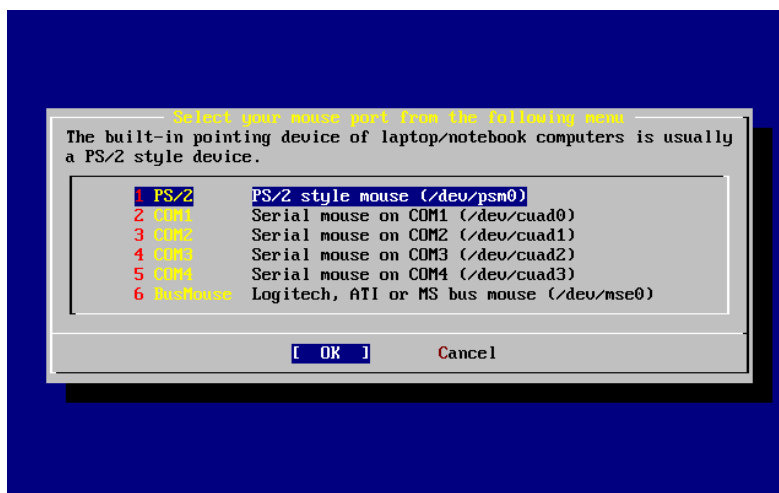
Myszka używana w przykładzie jest typu PS/2, wybrano więc domyślną opcję **Auto**. Inny protokół wybieramy wskazując odpowiednią opcję klawiszami kursora. Upewniwszy się, że [OK] jest zaznaczone, naciskamy **Enter** i wracamy do poprzedniego menu.

Rysunek 2-42. Konfiguracja portu myszki



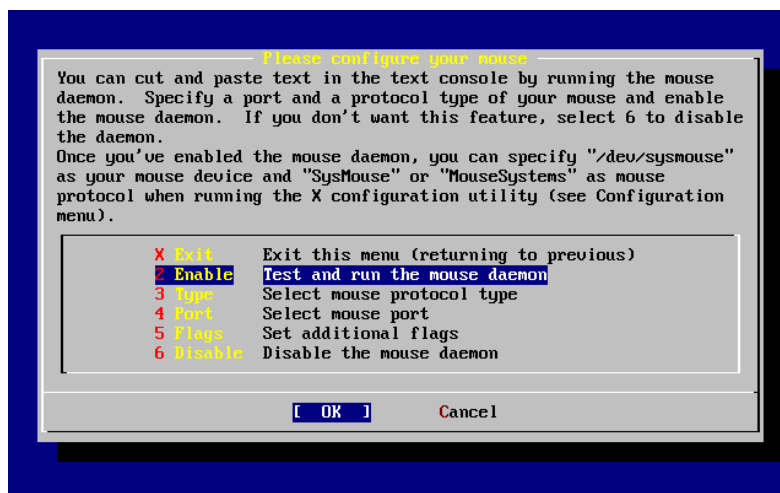
Za pomocą klawiszy kursora wybieramy Port i wciskamy **Enter**.

Rysunek 2-43. Wybór portu myszki



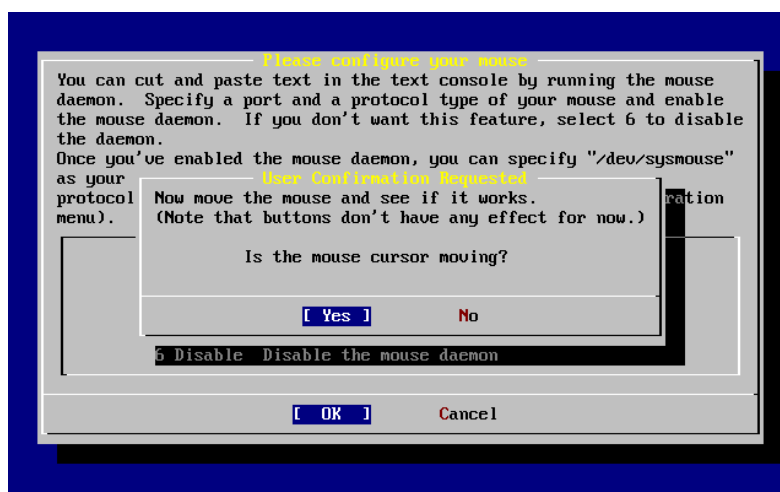
Ponieważ przykładowa myszka jest typu PS/2, zaznaczona została domyślna opcja PS/2. Klawiszami kursora możemy wybrać port, następnie naciskamy **Enter**.

Rysunek 2-44. Włączenie demona myszki



Na koniec wybieramy **Enable** i naciskamy **Enter** by włączyć demona myszki i go przetestować.

Rysunek 2-45. Testowanie demona myszki



Następnie musimy poruszyć myszką i sprawdzić czy kursor porusza się we właściwy sposób po ekranie. Jeśli tak to wybieramy [Yes] i wciskamy **Enter**. Jeśli nie myszka nie została właściwie skonfigurowana — wybieramy [No] i próbujemy innych ustawień myszy.

Wybieramy **Exit** i wciskamy **Enter**, by zakończyć ten etap konfiguracji.

2.9.11. Konfiguracja dodatkowych usług sieciowych

Napisał Tom Rhodes.

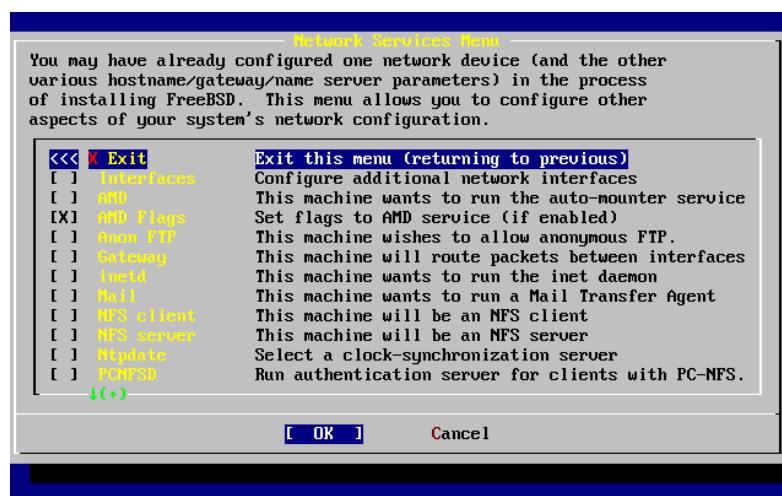
Konfiguracja usług sieciowych może być nużącym zadaniem dla początkujących użytkowników, szczególnie jeśli brak im wiedzy w tym zakresie. Możliwość pracy w sieci - także w Internecie - jest kluczowym elementem

wszystkich współczesnych systemów operacyjnych, w tym również FreeBSD. Stąd też jest bardzo pomocnym mieć pojęcie o możliwościach pracy w sieci jakie oferuje FreeBSD. Poznanie tych jego możliwości już w trakcie instalacji pozwoli użytkownikom zrozumieć różne aspekty funkcjonowania usług sieciowych.

Usługi sieciowe są programami potrafiącymi przyjmować dane z dowolnej lokalizacji w sieci. Dlatego właśnie dokładanych jest wiele starań, by zagwarantować, że programy te nie uczynią nic „szkodliwego”. Niestety, programiści nie są doskonali. W przeszłości zdarzały się sytuacje, w których atakujący wykorzystywali błędy w oprogramowaniu by wyrządzić szkodę systemowi. Stąd też jest bardzo istotnym by włączać tylko te usługi sieciowe, które są nam potrzebne. Jeśli nie jesteśmy pewni, najlepiej jest nie włączać danej usługi nim nie dowiemy się czy rzeczywiście jej potrzebujemy. Zawsze możemy ją aktywować później uruchamiając ponownie **sysinstall** bądź edytując plik `/etc/rc.conf`.

Wybranie opcji **Networking** spowoduje wyświetlenie menu zbliżonego do poniższego:

Rysunek 2-46. Najwyższy poziom konfiguracji sieci



Pierwszą z dostępnych opcji - **Interfaces** - opisuje bliżej Sekcja 2.9.1, dlatego też możemy ją teraz pominąć.

Wybór opcji **AMD** włączy wsparcie dla narzędzia automatycznego montowania BSD (ang. Automatic Mount Utility). Opcja ta najczęściej jest wykorzystywana z protokołem NFS (patrz poniżej) do automatycznego montowania zdalnych systemów plików. Nie wymaga dodatkowej konfiguracji.

Kolejną opcją jest **AMD Flags**. Po jej wybraniu pojawi się menu, gdzie należy wprowadzić specyficzne flagi AMD. Menu zawiera już domyślne wartości:

```
-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map
```

Flaga `-a` określa domyślny punkt montowania, w tym wypadku `/.amd_mnt`. Flaga `-l` definiuje domyślny plik log dziennika systemowego; jeśli w systemie wykorzystywany jest demon `syslogd`, wówczas wszystkie komunikaty będą wysyłane właśnie do niego. Katalog `/host` jest wykorzystywany do montowania systemów plików wyeksportowanych ze zdalnej maszyny, podczas gdy katalog `/net` do montowania systemów plików z adresu IP. Plik `/etc/amd.map` zawiera domyślne wartości flag dla zasobów eksportowanych przez AMD.

Wybór opcji **Anon FTP** zezwala na anonimowe połączenia FTP, tym samym tworząc z naszego komputera anonimowy serwer FTP. Należy mieć jednak świadomość niebezpieczeństw jakie pociąga za sobą taka konfiguracja.

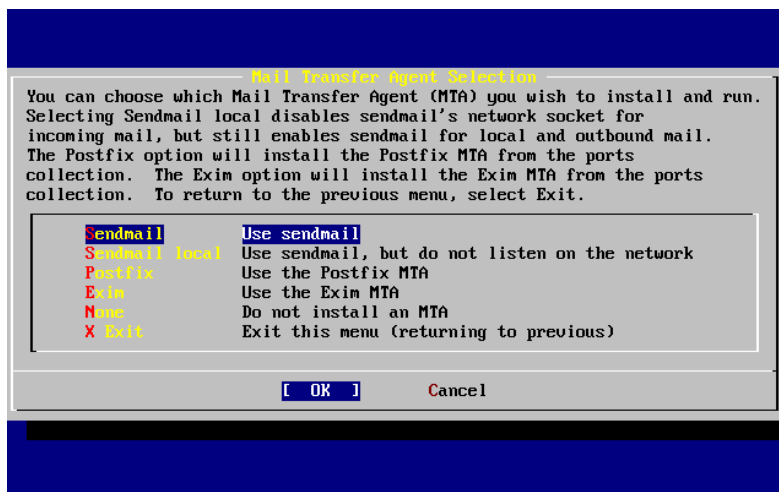
Po wybraniu tej opcji pojawi się kolejne okienko wyjaśniające związane z nią niebezpieczeństwa oraz umożliwiające szczegółową konfigurację.

Menu **Gateway** pozwala skonfigurować naszą maszynę jako bramę, co zostało opisane wcześniej. Może być również wykorzystane do wyłączenia tej opcji jeśli przypadkowo została ona aktywowana w trakcie instalacji.

Opcja **Inetd** pozwala skonfigurować bądź całkowicie wyłączyć demonona inetd(8), który również został opisany wcześniej.

Opcja **Mail** wykorzystywana jest do konfiguracji domyślnego systemowego serwera poczty MTA (ang. Mail Transfer Agent). Wybór tej opcji spowoduje wyświetlenie następującego menu:

Rysunek 2-47. Wybór domyślnego MTA



W menu tym mamy możliwość wyboru, który MTA zostanie zainstalowany jako domyślny. W praktyce MTA nie jest niczym więcej jak serwerem, który dostarcza pocztę elektroniczną do użytkowników lokalnego systemu bądź wysyła ją do Internetu.

Wybór opcji **Sendmail** spowoduje instalację popularnego serwera **sendmail**. Serwer ten jest domyślnym serwerem we FreeBSD. Opcja **Sendmail local** również spowoduje wybór **sendmail** jako domyślnego MTA, jednakże bez możliwości odbierania poczty przychodzącej z Internetu. Pozostałe opcje **Postfix** i **Exim** dają efekt analogiczny do **Sendmail** — obydwa rozwiązania dostarczają pocztę. Tym nie mniej, niektórzy użytkownicy preferują te serwery jako alternatywę dla MTA **sendmail**.

Po wybraniu MTA, bądź pominięciu tego kroku, pojawi się ponownie okno konfiguracji sieci z kolejną opcją: **NFS client**.

Opcja **NFS client** pozwala skonfigurować system do komunikacji z serwerem za pomocą NFS. Serwer NFS udostępnia systemy plików innym maszynom w sieci za pomocą protokołu NFS. Jeśli nasza maszyna nie będzie pracowała w sieci można tą opcję pominąć. System może później wymagać dalszej konfiguracji. Sekcja 25.3 zawiera szczegółowe informacje o konfiguracji klienta i serwera NFS.

Poniżej znajduje się opcja **NFS server** umożliwiająca skonfigurowanie systemu jako serwer NFS. Dodatkowo konfiguruje ona wymagane parametry dla usług RPC. RPC koordynuje połączenia pomiędzy maszynami i programami.

Kolejna opcja to Ntpdate, odpowiadająca za synchronizację czasu systemowego. Po wybraniu jej pojawi się następujące menu:

Rysunek 2-48. Konfiguracja ntpdate

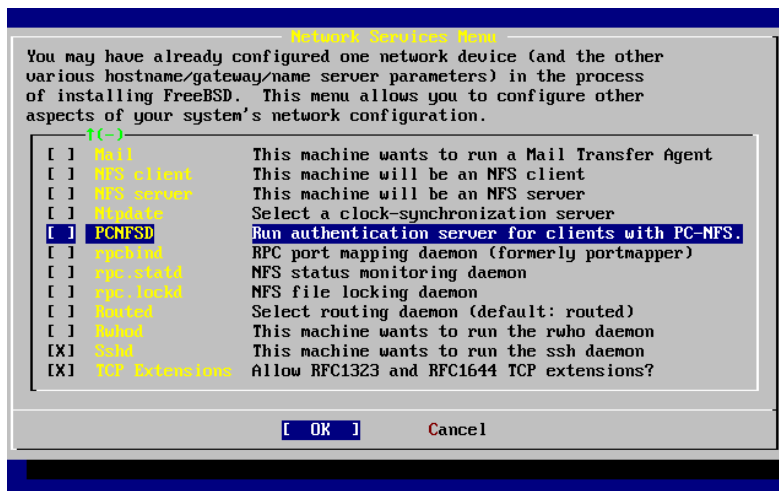


Z menu wybieramy najbliższy nam serwer. Wybór pobliskiego serwera gwarantuje dokładniejszą synchronizację czasu, z uwagi na fakt, że w komunikacji z bardziej oddalony serwerem mogą występować większe opóźnienia.

Kolejnym elementem jest wybór PCNFSD. Opcja ta zainstaluje net/pcnfsd z Kolekcji portów. Jest to przydatne narzędzie umożliwiające uwierzytelnianie NFS systemom operacyjnym, które same nie potrafią się uwierzytelić, jak np. MS-DOS.

Przewijając w dół pojawiają się kolejne opcje:

Rysunek 2-49. Najniższy poziom konfiguracji sieci



Programy rpcbind(8), rpc.statd(8) i rpc.lockd(8) wykorzystywane są przy połączeniach RPC (Remote Procedure Call). rpcbind zarządza komunikacją pomiędzy serwerem NFS i klientami, tym samym jest wymagany do

poprawnego funkcjonowania serwera NFS. Demon **rpc.statd** wykorzystywany jest do komunikacji z innymi demonami **rpc.statd** w sieci, w celu monitorowania stanu maszyn, na których one pracują. Uzyskane w ten sposób informacje przechowywane są z reguły w pliku `/var/db/statd.status`. Kolejnym elementem jest **rpc.lockd**, który udostępnia usługi blokowania plików. Z reguły, wykorzystywany jest w parze z **rpc.statd** do śledzenia, które maszyny wymagają blokowania i jak często. O ile dwie ostatnie usługi są idealne do debugowania, nie są one wymagane do poprawnego działania serwera NFS.

Kolejnym elementem na liście jest demon routowania - **Routed**. `routed(8)` zarządza tablicami routingu sieci, wyszukuje routery multicast i udostępnia na żądanie kopię tablic routingu każdej maszynie w sieci. Wykorzystywany jest on z reguły na komputerach pracujących jako bramy dla sieci lokalnej. Po jego wybraniu pojawi się dodatkowe menu, w którym należy określić jego domyślną lokalizację. Wartość domyślna jest zdefiniowana i zostanie wybrana po naciśnięciu klawisza **Enter**. Następnie pojawi się kolejne menu, tym razem w celu ustawienia flag. Domyślną jest `-q` i powinna pojawić się na ekranie.

Kolejną opcją jest **Rwhod**, której wybór włączy demona `rwhod(8)` w trakcie uruchamiania systemu. `rwhod` jest narzędziem, które regularnie rozsyła w sieci komunikaty systemowe bądź - w trybie „konsumenta” - zbiera je. Więcej informacji dostępnych jest w podręcznikach systemowych `ruptime(1)` i `rwho(1)`.

Przedostatnim elementem na liście jest demon `sshd(8)`. Jest to serwer **OpenSSH**, którego wykorzystanie jest zalecane w zamiast **telnetu** czy serwerów FTP. Serwer **sshd** jest wykorzystywany do zestawiania bezpiecznego połączenia pomiędzy dwoma maszynami wykorzystując połączenia szyfrowane.

Ostatnią na liście jest opcja Rozszerzeń TCP (TCP Extensions). Włączenie jej umożliwia korzystanie z rozszerzeń TCP zdefiniowanych w RFC 1323 i RFC 1644. O ile na wielu komputerach pozwoli to na przyspieszenie komunikacji, o tyle może również spowodować odrzucanie niektórych połączeń. Stosowanie tej opcji nie jest zalecane dla serwerów, choć może się okazać korzystne dla stacji roboczych.

Skończywszy konfigurację usług sieciowych możemy przewinąć do samej góry ekranu, do opcji **Exit** i przejść do kolejnej części konfiguracji.

2.9.12. Konfiguracja serwera X

Notatka: Począwszy od wersji FreeBSD 5.3-RELEASE, opcje konfiguracji serwera X zostały usunięte z **sysinstall**. Serwer X musimy zainstalować i skonfigurować po skończonej instalacji systemu. Rozdział 5 zawiera szczegółowe informacje odnośnie instalacji i konfiguracji serwera X. Jeśli nie instalujemy wersji wcześniejszej niż FreeBSD 5.3-RELEASE, możemy pominąć tą sekcję.

Chcąc korzystać z graficznego interfejsu użytkownika w rodzaju **KDE**, **GNIEME** lub innego, trzeba skonfigurować serwer X.

Notatka: By uruchomić **XFree86** z poziomu użytkownika innego niż `root`, należy zainstalować `x11/wrapper`. Jest on instalowany domyślnie we FreeBSD 4.7 i późniejszych. W przypadku wcześniejszych wersji można go zainstalować z menu wyboru pakietów.

Aby sprawdzić, czy nasza karta graficzna jest obsługiwana, możemy zajrzeć na stronę WWW XFree86 (<http://www.xfree86.org/>).

User Confirmation Requested

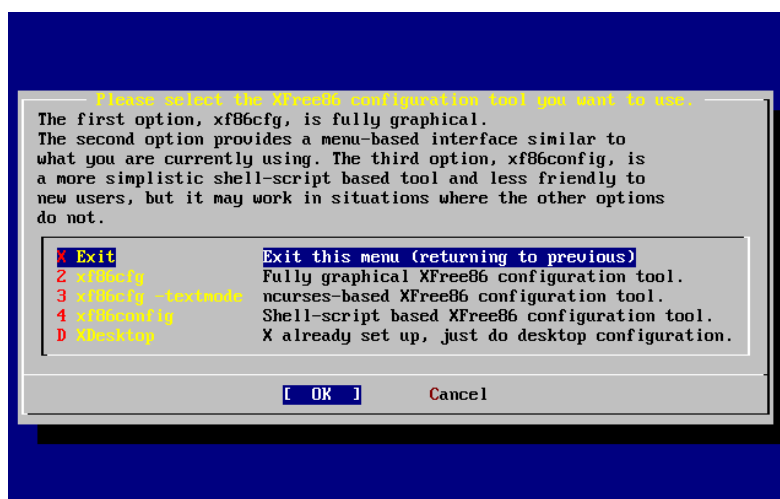
Would you like to configure your X server at this time?

[Yes] No

OstrzeżenieNależy koniecznie znać dane techniczne monitora i karty graficznej. Nieprawidłowe ustawienia mogą spowodować uszkodzenie sprzętu. Jeśli nie dysponujemy tymi danymi, wybierzmy [No] i przystąpmy do konfiguracji serwera X po zainstalowaniu systemu, gdy już zaopatrzymy się w niezbędne dane. Do tego celu możemy wykorzystać `sysinstall (/stand/sysinstall` we FreeBSD starszych niż 5.2), wybierając `Configure`, a następnie `XFree86`.

Jeśli mamy dane techniczne karty graficznej i monitora, wybieramy [Yes] i wciskamy **Enter**, rozpoczynając konfigurację serwera X.

Rysunek 2-50. Wybór metody konfiguracji



Serwer X można konfigurować na kilka sposobów. Wybieramy jedną z metod przy pomocy klawiszy kursora i naciskamy **Enter**. Pamiętajmy o uważnym czytaniu wszelkich poleceń pojawiających się na ekranie.

Wybór `xf86cfg` i `xf86cfg -textmode` może spowodować, że ekran stanie się ciemny, a uruchomienie może zająć kilka sekund. Bądźmy cierpliwi.

W poniższym przykładzie przedstawione będzie korzystanie z programu konfiguracyjnego `xf86config`. Wybierane przez nas opcje zależą będą od wyposażenia naszego komputera, będą się więc zapewne różnić od opcji pokazanych w przykładzie:

Message

You have configured and been running the mouse daemon.
 Choose `"/dev/sysmouse"` as the mouse port and `"SysMouse"` or `"MouseSystems"` as the mouse protocol in the X configuration utility.

[OK]

[Press enter to continue]

Komunikat ten informuje o wykryciu skonfigurowanego wcześniej demona myszki. Naciskamy **Enter**, by przejść dalej.

Po uruchomieniu, **xf86config** wyświetli krótkie wprowadzenie:

```
This program will create a basic XF86Config file, based on menu selections you make.
```

```
The XF86Config file usually resides in /usr/X11R6/etc/X11 or /etc/X11. A sample XF86Config file is supplied with XFree86; it is configured for a standard VGA card and monitor with 640x480 resolution. This program will ask for a pathname when it is ready to write the file.
```

```
You can either take the sample XF86Config as a base and edit it for your configuration, or let this program produce a base XF86Config file for your configuration and fine-tune it.
```

```
Before continuing with this program, make sure you know what video card you have, and preferably also the chipset it uses and the amount of video memory on your video card. SuperProbe may be able to help with this.
```

```
Press enter to continue, or ctrl-c to abort.
```

Po naciśnięciu **Enter** przejdziemy do konfiguracji myszki. Pamiętajmy, by uważnie czytać polecenia i wybrać właściwy protokół myszki „Mouse Systems” i port myszki `/dev/sysmouse`, nawet jeśli w przykładzie wybierana jest myszka PS/2.

First specify a mouse protocol type. Choose one from the following list:

1. Microsoft compatible (2-button protocol)
2. Mouse Systems (3-button protocol) & FreeBSD moused protocol
3. Bus Mouse
4. PS/2 Mouse
5. Logitech Mouse (serial, old type, Logitech protocol)
6. Logitech MouseMan (Microsoft compatible)
7. MM Series
8. MM HitTablet
9. Microsoft IntelliMouse

If you have a two-button mouse, it is most likely of type 1, and if you have a three-button mouse, it can probably support both protocol 1 and 2. There are two main varieties of the latter type: mice with a switch to select the protocol, and mice that default to 1 and require a button to be held at boot-time to select protocol 2. Some mice can be convinced to do 2 by sending a special sequence to the serial port (see the `ClearDTR/ClearRTS` options).

Enter a protocol number: 2

You have selected a Mouse Systems protocol mouse. If your mouse is normally in Microsoft-compatible mode, enabling the `ClearDTR` and `ClearRTS` options may cause it to switch to Mouse Systems mode when the server starts.

Please answer the following question with either 'y' or 'n'.

Do you want to enable ClearDTR and ClearRTS? n

You have selected a three-button mouse protocol. It is recommended that you do not enable Emulate3Buttons, unless the third button doesn't work.

Please answer the following question with either 'y' or 'n'.

Do you want to enable Emulate3Buttons? y

Now give the full device name that the mouse is connected to, for example /dev/tty00. Just pressing enter will use the default, /dev/mouse. On FreeBSD, the default is /dev/sysmouse.

Mouse device: /dev/sysmouse

Kolejnym krokiem jest konfiguracja klawiatury. W przykładzie wybrana została typowa klawiatura o 101 klawiszach. Jako wariant nazwy możemy wybrać dowolną nazwę, lub po prostu nacisnąć **Enter**, akceptując proponowaną nazwę domyślną.

Please select one of the following keyboard types that is the better description of your keyboard. If nothing really matches, choose 1 (Generic 101-key PC)

- 1 Generic 101-key PC
- 2 Generic 102-key (Intl) PC
- 3 Generic 104-key PC
- 4 Generic 105-key (Intl) PC
- 5 Dell 101-key PC
- 6 Everex STEPnote
- 7 Keytronic FlexPro
- 8 Microsoft Natural
- 9 Northgate OmniKey 101
- 10 Winbook Model XP5
- 11 Japanese 106-key
- 12 PC-98xx Series
- 13 Brazilian ABNT2
- 14 HP Internet
- 15 Logitech iTouch
- 16 Logitech Cordless Desktop Pro
- 17 Logitech Internet Keyboard
- 18 Logitech Internet Navigator Keyboard
- 19 Compaq Internet
- 20 Microsoft Natural Pro
- 21 Genius Comfy KB-16M
- 22 IBM Rapid Access
- 23 IBM Rapid Access II
- 24 Chicony Internet Keyboard
- 25 Dell Internet Keyboard

Enter a number to choose the keyboard.

1

Please select the layout corresponding to your keyboard

- 1 U.S. English
- 2 U.S. English w/ ISO9995-3
- 3 U.S. English w/ deadkeys
- 4 Albanian
- 5 Arabic
- 6 Armenian
- 7 Azerbaidjani
- 8 Belarusian
- 9 Belgian
- 10 Bengali
- 11 Brazilian
- 12 Bulgarian
- 13 Burmese
- 14 Canadian
- 15 Croatian
- 16 Czech
- 17 Czech (qwerty)
- 18 Danish

Enter a number to choose the country.

Press enter for the next page

1

Please enter a variant name for 'us' layout. Or just press enter for default variant

us

Please answer the following question with either 'y' or 'n'.

Do you want to select additional XKB options (group switcher, group indicator, etc.)? n

Następnie przystępujemy do konfiguracji monitora. Pamiętajmy, by nie przekroczyć dopuszczalnych wartości częstotliwości, ponieważ może to spowodować uszkodzenie monitora. W razie jakichkolwiek wątpliwości, odłóżmy konfigurację monitora do czasu, gdy będziemy już mieć niezbędne informacje.

Now we want to set the specifications of the monitor. The two critical parameters are the vertical refresh rate, which is the rate at which the whole screen is refreshed, and most importantly the horizontal sync rate, which is the rate at which scanlines are displayed.

The valid range for horizontal sync and vertical sync should be documented in the manual of your monitor. If in doubt, check the monitor database /usr/X11R6/lib/X11/doc/Monitors to see if your monitor is there.

Press enter to continue, or ctrl-c to abort.

You must indicate the horizontal sync range of your monitor. You can either select one of the predefined ranges below that correspond to industry-standard monitor types, or give a specific range.

It is VERY IMPORTANT that you do not specify a monitor type with a horizontal sync range that is beyond the capabilities of your monitor. If in doubt, choose a conservative setting.

```

hsync in kHz; monitor type with characteristic modes
1  31.5; Standard VGA, 640x480 @ 60 Hz
2  31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
3  31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no 800x600)
4  31.5, 35.15, 35.5; Super VGA, 1024x768 @ 87 Hz interlaced, 800x600 @ 56 Hz
5  31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
6  31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
7  31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
8  31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
9  31.5 - 79.0; Monitor that can do 1280x1024 @ 74 Hz
10 31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
11 Enter your own horizontal sync range

```

Enter your choice (1-11): 6

You must indicate the vertical sync range of your monitor. You can either select one of the predefined ranges below that correspond to industry-standard monitor types, or give a specific range. For interlaced modes, the number that counts is the high one (e.g. 87 Hz rather than 43 Hz).

```

1  50-70
2  50-90
3  50-100
4  40-150
5  Enter your own vertical sync range

```

Enter your choice: 2

You must now enter a few identification/description strings, namely an identifier, a vendor name, and a model name. Just pressing enter will fill in default names.

The strings are free-form, spaces are allowed.

Enter an identifier for your monitor definition: Hitachi

W kolejnym etapie wybieramy z listy sterownik karty graficznej. Jeśli przewijając listę niechcący ominiemy naszą kartę, naciskajmy dalej **Enter**, a lista zostanie powtórzona. W przykładzie pokazujemy tylko fragment listy:

Now we must configure video card specific settings. At this point you can choose to make a selection out of a database of video card definitions. Because there can be variation in Ramdacs and clock generators even between cards of the same model, it is not sensible to blindly copy the settings (e.g. a Device section). For this reason, after you make a

selection, you will still be asked about the components of the card, with the settings from the chosen database entry presented as a strong hint.

The database entries include information about the chipset, what driver to run, the Ramdac and ClockChip, and comments that will be included in the Device section. However, a lot of definitions only hint about what driver to run (based on the chipset the card uses) and are untested.

If you can't find your card in the database, there's nothing to worry about. You should only choose a database entry that is exactly the same model as your card; choosing one that looks similar is just a bad idea (e.g. a GemStone Snail 64 may be as different from a GemStone Snail 64+ in terms of hardware as can be).

Do you want to look at the card database? y

288	Matrox Millennium G200 8MB	mgag200
289	Matrox Millennium G200 SD 16MB	mgag200
290	Matrox Millennium G200 SD 4MB	mgag200
291	Matrox Millennium G200 SD 8MB	mgag200
292	Matrox Millennium G400	mgag400
293	Matrox Millennium II 16MB	mga2164w
294	Matrox Millennium II 4MB	mga2164w
295	Matrox Millennium II 8MB	mga2164w
296	Matrox Mystique	mga1064sg
297	Matrox Mystique G200 16MB	mgag200
298	Matrox Mystique G200 4MB	mgag200
299	Matrox Mystique G200 8MB	mgag200
300	Matrox Productiva G100 4MB	mgag100
301	Matrox Productiva G100 8MB	mgag100
302	MediaGX	mediagx
303	MediaVision Proaxcel 128	ET6000
304	Mirage Z-128	ET6000
305	Miro CRYSTAL VRX	Verite 1000

Enter a number to choose the corresponding card definition.
Press enter for the next page, q to continue configuration.

288

Your selected card definition:

```
Identifier: Matrox Millennium G200 8MB
Chipset:    mgag200
Driver:     mga
Do NIET probe clocks or use any Clocks line.
```

Press enter to continue, or ctrl-c to abort.

Now you must give information about your video card. This will be used for the "Device" section of your video card in XF86Config.

You must indicate how much video memory you have. It is probably a good idea to use the same approximate amount as that detected by the server you intend to use. If you encounter problems that are due to the used server not supporting the amount memory you have (e.g. ATI Mach64 is limited to 1024K with the SVGA server), specify the maximum amount supported by the server.

How much video memory do you have on your video card:

- 1 256K
- 2 512K
- 3 1024K
- 4 2048K
- 5 4096K
- 6 Other

Enter your choice: 6

Amount of video memory in Kbytes: 8192

You must now enter a few identification/description strings, namely an identifier, a vendor name, and a model name. Just pressing enter will fill in default names (possibly from a card definition).

Your card definition is Matrox Millennium G200 8MB.

The strings are free-form, spaces are allowed.

Enter an identifier for your video card definition:

Następnie wybieramy tryby graficzne dla preferowanych rozdzielczości. Najczęściej używane są tryby 640x480, 800x600 i 1024x768, wybór zależy jednak od możliwości karty graficznej, rozmiarów monitora i oczekiwanej wygody pracy. Gdy będziemy wybierać głębię koloru, wybierzmy najwyższą wartość, którą obsługuje karta.

For each depth, a list of modes (resolutions) is defined. The default resolution that the server will start-up with will be the first listed mode that can be supported by the monitor and card. Currently it is set to:

```
"640x480" "800x600" "1024x768" "1280x1024" for 8-bit
"640x480" "800x600" "1024x768" "1280x1024" for 16-bit
"640x480" "800x600" "1024x768" "1280x1024" for 24-bit
```

Modes that cannot be supported due to monitor or clock constraints will be automatically skipped by the server.

- 1 Change the modes for 8-bit (256 colors)
- 2 Change the modes for 16-bit (32K/64K colors)
- 3 Change the modes for 24-bit (24-bit color)
- 4 The modes are OK, continue.

Enter your choice: 2

Select modes from the following list:

- 1 "640x400"
- 2 "640x480"
- 3 "800x600"
- 4 "1024x768"
- 5 "1280x1024"
- 6 "320x200"
- 7 "320x240"
- 8 "400x300"
- 9 "1152x864"
- a "1600x1200"
- b "1800x1400"
- c "512x384"

Please type the digits corresponding to the modes that you want to select.
For example, 432 selects "1024x768" "800x600" "640x480", with a
default mode of 1024x768.

Which modes? 432

You can have a virtual screen (desktop), which is screen area that is larger
than the physical screen and which is panned by moving the mouse to the edge
of the screen. If you don't want virtual desktop at a certain resolution,
you cannot have modes listed that are larger. Each color depth can have a
differently-sized virtual screen

Please answer the following question with either 'y' or 'n'.
Do you want a virtual screen that is larger than the physical screen? n

For each depth, a list of modes (resolutions) is defined. The default
resolution that the server will start-up with will be the first listed
mode that can be supported by the monitor and card.
Currently it is set to:

"640x480" "800x600" "1024x768" "1280x1024" for 8-bit
"1024x768" "800x600" "640x480" for 16-bit
"640x480" "800x600" "1024x768" "1280x1024" for 24-bit

Modes that cannot be supported due to monitor or clock constraints will
be automatically skipped by the server.

- 1 Change the modes for 8-bit (256 colors)
- 2 Change the modes for 16-bit (32K/64K colors)
- 3 Change the modes for 24-bit (24-bit color)
- 4 The modes are OK, continue.

Enter your choice: 4

Please specify which color depth you want to use by default:

- 1 1 bit (monochrome)
- 2 4 bits (16 colors)
- 3 8 bits (256 colors)
- 4 16 bits (65536 colors)
- 5 24 bits (16 million colors)

Enter a number to choose the default depth.

4

Przygotowaną konfigurację należy zachować. Upewnijmy się, że konfiguracja zostanie zapisana w pliku o nazwie `/etc/X11/XF86Config`.

I am going to write the XF86Config file now. Make sure you don't accidentally overwrite a previously configured one.

Shall I write it to `/etc/X11/XF86Config`? y

Jeśli z jakichś przyczyn konfiguracja nie powiedzie się, możemy zacząć ją od początku, wybierając [Yes], gdy pojawi się następujący komunikat:

```
User Confirmation Requested
The XFree86 configuration process seems to have
failed. Would you like to try again?
```

```
[ Yes ]      No
```

Jeżeli konfiguracja **XFree86** sprawia problemy, wybierzmy [No] i naciśnijmy **Enter**, by kontynuować instalację. Po jej zakończeniu będziemy mogli uruchomić program konfiguracyjny poleceniem `xf86cfg -textmode` lub `xf86config`, wydany jako `root`. Rozdział 5 prezentuje inną metodę konfiguracji **XFree86**. Jeśli zdecydujemy się pominąć na razie konfigurację **XFree86**, kolejnym krokiem będzie wybór pakietów.

Domyślnie serwer X może zostać unicestwiony kombinacją klawiszy **Ctrl+Alt+Backspace**. Możemy z niej skorzystać, jeśli coś jest nie w porządku z ustawieniami serwera i chcemy uniknąć uszkodzenia sprzętu.

Podczas pracy serwera X można zmieniać tryb graficzny, używając kombinacji klawiszy **Ctrl+Alt++** lub **Ctrl+Alt+-**.

Po zakończeniu instalacji można wyregulować wysokość, szerokość i położenie obrazu przy użyciu **xvidtune**, po uruchomieniu **XFree86**.

Zwracamy uwagę na ostrzeżenia o możliwości uszkodzenia sprzętu poprzez niewłaściwe ustawienia. Nie róbmy niczego, czego nie jesteśmy pewni. Zamiast używać **xvidtune**, możemy dostroić ekran X Window korzystając z regulatorów monitora. Mogą się pojawić pewne różnice w wyświetlaniu obrazu przy powrocie do trybu tekstowego, lepsze to jednak niż uszkodzenie sprzętu.

Przed dokonaniem jakichkolwiek zmian zapoznajmy się z dokumentacją **xvidtune(1)**.

Jeżeli konfiguracja **XFree86** przebiegła pomyślnie, przejdziemy do kolejnego etapu, w którym wybierzemy menedżera okien.

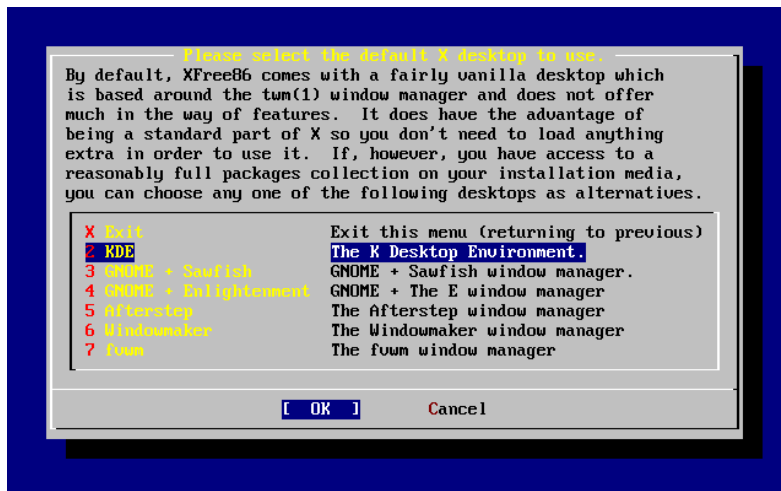
2.9.13. Wybór menedżera okien

Notatka: Począwszy od wersji FreeBSD 5.3-RELEASE, opcje wyboru środowiska graficznego zostały usunięte z `sysinstall`. Musimy je skonfigurować po skończonej instalacji systemu. Rozdział 5 zawiera szczegółowe informacje odnośnie instalacji i konfiguracji środowiska graficznego. Jeśli nie instalujemy wersji wcześniejszej niż FreeBSD 5.3-RELEASE, możemy pominąć tą sekcję.

Dostępnych jest wiele różnych menedżerów okien, poczynając od najprostszych, zapewniających jedynie podstawowe funkcje, do rozbudowanych środowisk wyposażonych w pokaźny zestaw oprogramowania. Niektórym wystarczy nieznaczna przestrzeń na dysku i niewiele pamięci, inne natomiast mogą mieć znacznie większe wymagania. Dobrze jest wypróbować kilka różnych menedżerów i wybrać spośród nich ten, który najbardziej nam odpowiada. Są one dostępne w Kolekcji portów lub w postaci pakietów, można je więc instalować po zainstalowaniu systemu.

Możemy wybrać jeden z popularnych menedżerów okien i zainstalować go jako domyślny. Dzięki temu będziemy mieć możliwość uruchomienia go zaraz po zakończeniu instalacji.

Rysunek 2-51. Wybór domyślnego menedżera okien



Klawiszami kursora wybieramy jedną z opcji i wciskamy **Enter**. Wybrany menedżer okien zostanie zainstalowany.

2.9.14. Instalacja pakietów

Pakiety to skompilowane programy, które można w łatwy sposób instalować.

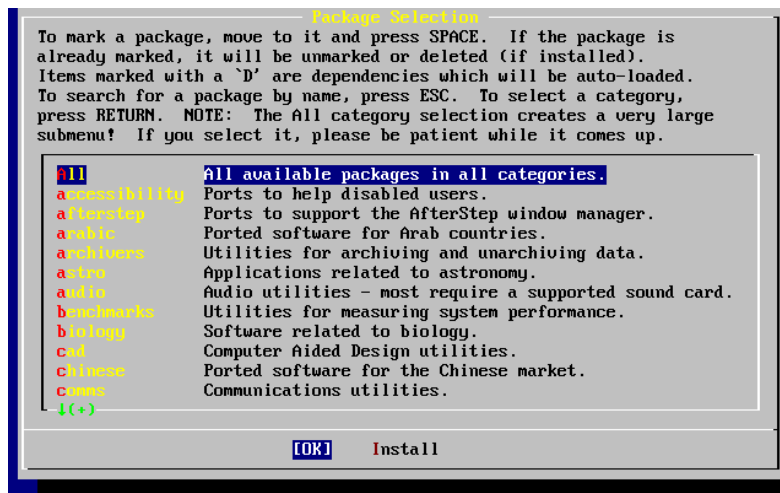
W poniższym przykładzie pokazana jest instalacja jednego pakietu. Możemy oczywiście zainstalować więcej pakietów. Gdy system będzie już zainstalowany, kolejne pakiety będzie można dodawać przy użyciu `sysinstall` (`/stand/sysinstall` w wersjach FreeBSD wcześniejszych niż 5.2).

```
User Confirmation Requested
The FreeBSD package collection is a collection of hundreds of
ready-to-run applications, from text editors to games to WEB servers
and more. Would you like to browse the collection now?
```

[Yes] No

Jeśli wybierzemy [Yes] i naciśniemy **Enter**, przejdziemy do ekranu wyboru pakietów:

Rysunek 2-52. Wybór kategorii pakietów

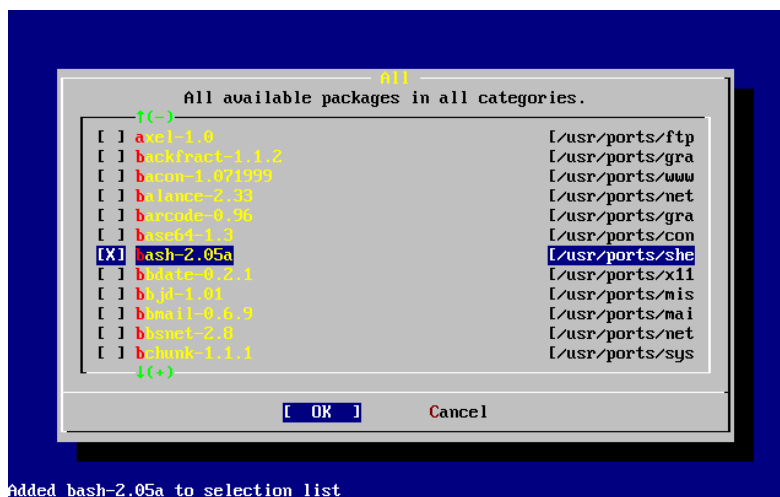


W danej chwili dostępne do instalacji są jedynie pakiety z bieżącego nośnika.

Możemy wybrać jedną z kategorii pakietów albo All, by wyświetlone zostały wszystkie dostępne pakiety. Wybraną opcję wskazujemy przy użyciu klawiszy kursora i wciskamy **Enter**.

Pokazana zostanie lista pakietów dostępnych w wybranej kategorii:

Rysunek 2-53. Wybór pakietów



Added bash-2.05a to selection list

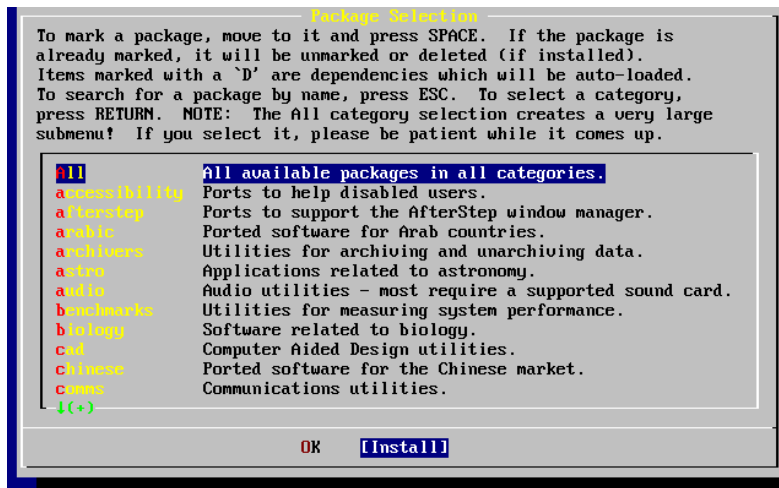
Dla przykładu zaznaczona została powłoka **bash**. Możemy wybrać tyle pakietów, ile nam się podoba, zaznaczając każdy z nich **Space**. Krótki opis pakietu wyświetlany jest w lewym dolnym rogu ekranu.

Klawiszem **Tab** możemy przełączać się między ostatnio wybranym pakietem, przyciskami [OK] i [Cancel].

Po zaznaczeniu wszystkich wybranych pakietów naciskamy **Tab**, by zaznaczyć [OK] i naciskamy **Enter**, powracając w ten sposób do menu wyboru pakietów.

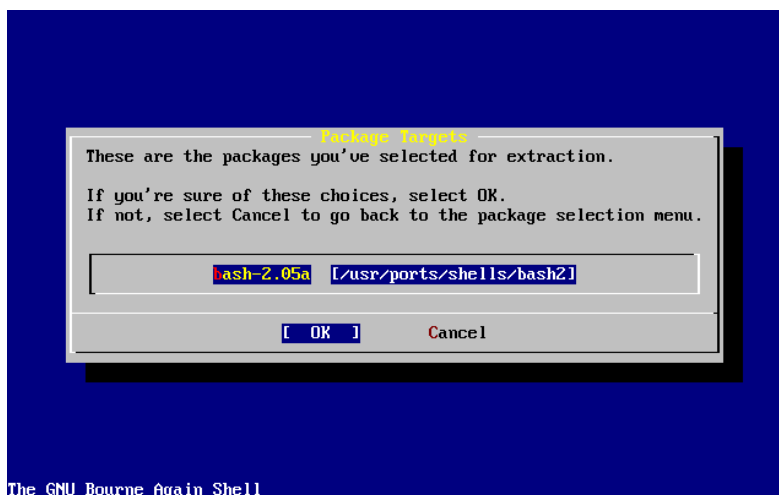
Do przełączania się między [OK] i [Cancel] mogą również służyć klawisze kursora. Za ich pomocą możemy wybrać [OK], a następnie nacisnąć **Enter**, by wrócić do menu wyboru pakietów.

Rysunek 2-54. Rozpoczęcie instalacji pakietów



Klawiszami kursora i **Tab** wybieramy [Install] i wciskamy **Enter**. Pojawi się prośba o potwierdzenie chęci zainstalowania pakietów:

Rysunek 2-55. Potwierdzenie instalacji pakietów



Gdy wybierzemy [OK] i naciśniemy **Enter**, rozpocznie się instalacja pakietów. Aż do jej zakończenia będą pokazywane komunikaty o przebiegu instalacji. Jeżeli pojawią się informacje o jakichkolwiek problemach, zanotujmy je.

Po zainstalowaniu pakietów wracamy do konfiguracji systemu. Nawet jeśli nie wybraliśmy żadnych pakietów i chcemy wrócić do końcowej konfiguracji wybieramy opcję **Install**.

2.9.15. Dodawanie użytkowników i grup

Powinniśmy założyć przynajmniej jedno konto użytkownika, by móc korzystać z systemu nie będąc zalogowanym jako `root`. Główna partycja jest zwykle niewielka, więc korzystanie z aplikacji jako `root` może ją szybko zappełnić. Inny powód wymieniony został w poniższym komunikacie:

```

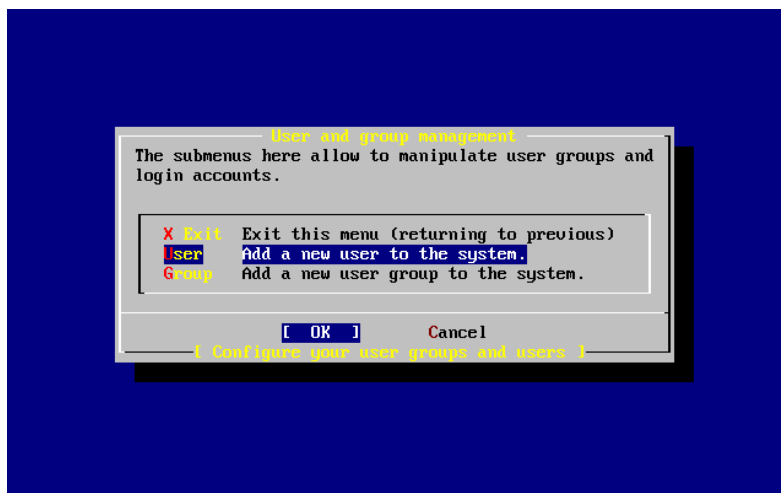
User Confirmation Requested

Would you like to add any initial user accounts to the system? Adding
at least one account for yourself at this stage is suggested since
working as the "root" user is dangerous (it is easy to do things which
adversely affect the entire system).
```

```
[ Yes ]   No
```

Wybieramy `[Yes]` i naciskamy **Enter**, by dodać użytkownika.

Rysunek 2-56. Dodawanie użytkownika



Klawiszamy kursora wybieramy **User** (użytkownik) i wciskamy **Enter**.

Rysunek 2-57. Dane nowego użytkownika

Kolejne pola wybieramy klawiszem **Tab**. W dolnej części ekranu pojawiać się będą następujące opisy, pomocne przy wprowadzaniu poszczególnych danych:

Login ID

Nazwa nowego użytkownika (obowiązkowa).

UID

Numer będący identyfikatorem użytkownika (wypełniany automatycznie, jeśli pole pozostanie puste).

Group

Nazwa podstawowej grupy użytkownika (wybierana automatycznie, jeśli pole pozostanie puste).

Password

Hasło użytkownika (wpisujemy je uważnie!).

Full name

Nazwisko użytkownika (komentarz).

Member groups

Grupy, których członkiem będzie użytkownik (czyli dostanie ich uprawnienia).

Home directory

Domowy katalog użytkownika (wpisywany automatycznie, jeśli pole pozostanie puste).

Login shell

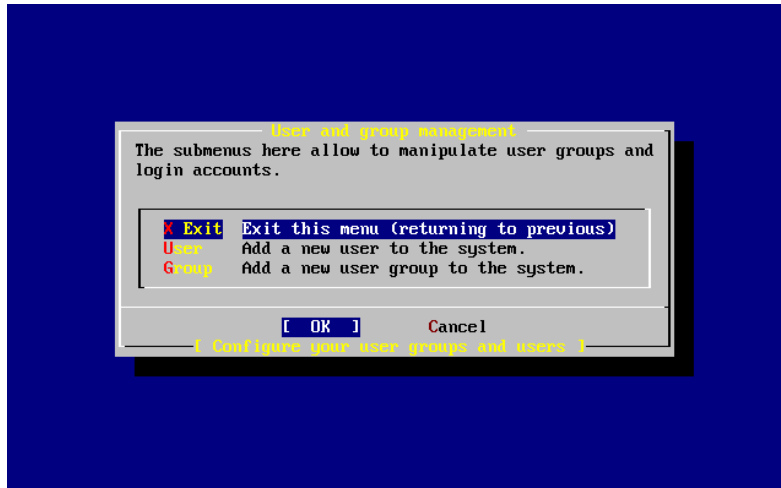
Powłoka uruchamiana po zalogowaniu się (wybierana automatycznie, jeśli pole pozostanie puste, np. `/bin/sh`).

W przykładzie powłoka została zmieniona z `/bin/sh` na `/usr/local/bin/bash`, aby korzystać z powłoki **bash** zainstalowanej wcześniej jako pakiet. Nie wpisujemy tu powłoki, która nie istnieje, gdyż uniemożliwi to zalogowanie się. Najpopularniejszą powłoką w świecie BSD jest powłoka C, czyli `/bin/tcsh`.

Użytkownik został dopisany do grupy `wheel`, dzięki czemu będzie mógł uzyskiwać uprawnienia użytkownika `root`.

Gdy skończymy, wybieramy [OK]. Ponownie pojawi się menu zarządzania użytkownikami i grupami:

Rysunek 2-58. Wyjście z menu zarządzania użytkownikami i grupami



W podobny sposób możemy od razu utworzyć dodatkowe grupy, jeśli zajdzie taka potrzeba. Gdy system będzie już zainstalowany, będziemy mogli dodawać grupy przy użyciu `sysinstall` (`/stand/sysinstall` w wersjach FreeBSD starszych niż 5.2).

Gdy skończymy dodawanie użytkowników wybieramy klawiszami kursora **Exit** i wciskamy **Enter**, by kontynuować instalację.

2.9.16. Hasło użytkownika `root`

Message

Now you must set the system manager's password.

This is the password you'll use to log in as "root".

[OK]

[Press enter to continue]

Wciskamy **Enter**, aby ustawić hasło `roota`.

Hasło musi być prawidłowo podane dwukrotnie. Rzecz jasna, powinniśmy zadbać o to, by łatwo odnaleźć hasło, gdy zdarzy się nam je zapomnieć. Zwróćmy uwagę, że w trakcie wpisywania hasła nie pojawiają się żadne znaki, nawet gwiazdki.

Changing local password for root.

New password :

Retype new password :

Po pomyślnym wprowadzeniu hasła przejdziemy do kolejnego etapu instalacji.

2.9.17. Zakończenie instalacji

Jeżeli będziemy chcieli skonfigurować dodatkowe urządzenia sieciowe, lub wprowadzić inne zmiany w konfiguracji systemu, możemy to zrobić w tym właśnie momencie, lub też po zakończeniu instalacji za pośrednictwem `sysinstall (/stand/sysinstall` w wersjach FreeBSD wcześniejszych niż 5.2).

```

User Confirmation Requested
Visit the general configuration menu for a chance to set any last
options?

```

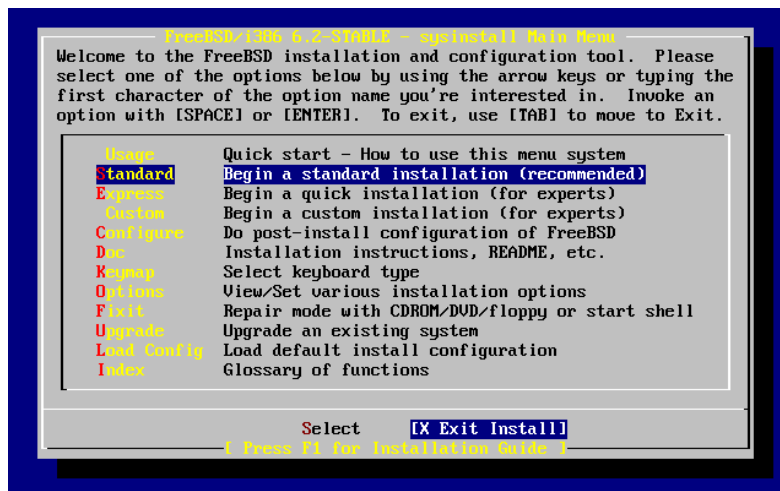
```

Yes      [ No ]

```

Wybieramy klawiszami kursora [No] i wciskamy **Enter**, by powrócić do głównego menu instalacji.

Rysunek 2-59. Zakończenie instalacji



Przy pomocy klawiszy kursora wybieramy [X Exit Install] i naciskamy **Enter**. Pojawi się prośba o potwierdzenie chęci zakończenia instalacji:

```

User Confirmation Requested
Are you sure you wish to exit? The system will reboot (be sure to
remove any floppies from the drives).

```

```

[ Yes ]   No

```

Wybieramy [Yes]. Jeżeli uruchamialiśmy komputer z dyskietki, wyjmujemy ją. Napęd CDROM będzie zablokowany aż do chwili, gdy komputer zacznie się ponownie uruchamiać. Wtedy napęd zostanie odblokowany i będzie można wyjąć z niego płytę (szybko).

Komputer zostanie ponownie uruchomiony. Zwróćmy uwagę na ewentualne komunikaty o błędach.

2.9.18. Uruchamianie FreeBSD

2.9.18.1. Uruchamianie FreeBSD na komputerach i386

Jeżeli wszystko przebiegło prawidłowo, na ekranie zobaczymy serię kolejno pojawiających się komunikatów, a na koniec będziemy mogli się zalogować. Komunikaty możemy przeczytać naciskając **Scroll-Lock**, następnie przewijając ekran klawiszami **PgUp** i **PgDn**. Ponownie naciskając **Scroll-Lock** powracamy do komunikatu logowania.

Być może nie będziemy mogli zobaczyć wszystkich komunikatów (ograniczony rozmiar bufora), jednak można je przejrzeć po zalogowaniu się, wpisując `dmesg` w linii poleceń.

Zalogujmy się, wpisując nazwę użytkownika i hasło wybrane podczas instalacji (w naszym przykładzie `rpratt`). Jako `root` powinniśmy logować się tylko wtedy, gdy jest to konieczne.

Typowe komunikaty pokazywane podczas uruchamiania systemu (pominięto informacje o wersji):

```
Copyright (c) 1992-2002 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
    The Regents of the University of California. All rights reserved.
```

```
Timecounter "i8254" frequency 1193182 Hz
CPU: AMD-K6(tm) 3D processor (300.68-MHz 586-class CPU)
  Origin = "AuthenticAMD" Id = 0x580 Stepping = 0
  Features=0x8001bf<FPU,VME,DE,PSE,TSC,MSR,MCE,CX8,MMX>
  AMD Features=0x80000800<SYSCALL,3DNow!>
real memory = 268435456 (262144K bytes)
config> di sn0
config> di lnc0
config> di le0
config> di ie0
config> di fe0
config> di cs0
config> di bt0
config> di aic0
config> di aha0
config> di adv0
config> q
avail memory = 256311296 (250304K bytes)
Preloaded elf kernel "kernel" at 0xc0491000.
Preloaded userconfig_script "/boot/kernel.conf" at 0xc049109c.
md0: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1: <VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <ISA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
```

```

atal: at 0x170 irq 15 on atapci0
uhci0: <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci0
usb0: <VIA 83C572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr 1
uhub0: 2 ports with 2 removable, self powered
chip1: <VIA 82C586B ACPI interface> at device 7.3 on pci0
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xe800-0xe81f irq 9 at
device 10.0 on pci0
ed0: address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq 2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbd0: <keyboard controller (i8042)> at port 0x60-0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq 1 on atkbd0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbd0
psm0: model Generic PS/2 mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x1 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
ppbus0: IEEE1284 device found /NIBBLE
Probing for PnP devices on ppbus0:
plip0: <PLIP network interface> on ppbus0
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
ppi0: <Parallel I/O> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master using UDMA33
ad2: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata1-master using UDMA33
acd0: CDROM <DELTA OTC-H101/ST3 F/W by OIPD> at ata0-slave using PIO4
Mounting root from ufs:/dev/ad0sla
swapon: adding /dev/ad0slb as swap device
Automatic boot in progress...
/dev/ad0sla: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0sla: clean, 48752 free (552 frags, 6025 blocks, 0.9% fragmentation)
/dev/ad0slf: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0slf: clean, 128997 free (21 frags, 16122 blocks, 0.0% fragmentation)
/dev/ad0slg: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0slg: clean, 3036299 free (43175 frags, 374073 blocks, 1.3% fragmentation)
/dev/ad0sle: filesystem CLEAN; SKIPPING CHECKS
/dev/ad0sle: clean, 128193 free (17 frags, 16022 blocks, 0.0% fragmentation)
Doing initial network setup: hostname.
ed0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.0.1 netmask 0xfffff00 broadcast 192.168.0.255

```

```

        inet6 fe80::5054::5ff::fede:731b%ed0 prefixlen 64 tentative scopeid 0x1
        ether 52:54:05:de:73:1b
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x8
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
Additional routing options: IP gateway=TAK TCP keepalive=TAK
routing daemons:.
additional daemons: syslogd.
Doing additional network setup:.
Starting final network daemons: creating ssh RSA host key
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
cd:76:89:16:69:0e:d0:6e:f8:66:d0:07:26:3c:7e:2d root@k6-2.example.com
creating ssh DSA host key
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
f9:a1:a9:47:c4:ad:f9:8d:52:b8:b8:ff:8c:ad:2d:e6 root@k6-2.example.com.
setting ELF ldconfig path: /usr/lib /usr/lib/compat /usr/X11R6/lib
/usr/local/lib
a.out ldconfig path: /usr/lib/aout /usr/lib/compat/aout /usr/X11R6/lib/aout
starting standard daemons: inetd cron sshd usbd sendmail.
Initial rc.i386 initialization:.
rc.i386 configuring syscons: blank_time screensaver moused.
Additional ABI support: linux.
Local package initialization:.
Additional TCP options:.

FreeBSD/i386 (k6-2.example.com) (ttyv0)

login: rpratt
Password:

```

Generowanie kluczy RSA i DSA na niezbyt szybkich komputerach może zająć nieco czasu. Dzieje się to tylko podczas pierwszego uruchomienia nowo zainstalowanego systemu. Następne ładowanie systemu będzie już odbywać się szybciej.

Jeśli skonfigurowaliśmy serwer X i wybraliśmy menedżera okien, możemy uruchomić go wpisując `startx` w linii poleceń.

2.9.18.2. Uruchamianie FreeBSD na komputerach Alpha

Po zakończeniu instalacji będziemy mogli uruchomić FreeBSD, wpisując następujące polecenie w konsoli SRM:

```
>>>BOOT DKC0
```

Nakazuje ono oprogramowaniu sprzętowemu uruchomić system z określonego dysku. By FreeBSD było automatycznie uruchamiane przy włączeniu komputera, wpisujemy poniższe polecenia:

```
>>> SET BOOT_OSFLAGS A
>>> SET BOOT_FILE "
>>> SET BOOTDEF_DEV DKC0
>>> SET AUTO_ACTION BOOT
```

Komunikaty pokazywane podczas ładowania systemu będą podobne (choć nie identyczne) do komunikatów pokazywanych na i386.

2.9.19. Wyłączanie FreeBSD

Właściwe wyłączenie systemu operacyjnego jest istotną sprawą. Nie należy po prostu wyłączać komputera. Powinniśmy najpierw uzyskać prawa administratora, wpisując w linii poleceń `su` i podając hasło `roota`; może to zrobić tylko użytkownik należący do grupy `wheel`. Możemy także po prostu zalogować się jako `root`. Następnie wydajemy polecenie `shutdown -h now`.

```
The operating system has halted.
Please press any key to reboot.
```

Po takim wyłączeniu systemu i pojawieniu się komunikatu „Please press any key to reboot” (Naciśnij dowolny klawisz by ponownie uruchomić system), można już wyłączyć komputer. Naciśnięcie dowolnego klawisza spowoduje ponowne uruchomienie systemu.

Inny sposobem ponownego uruchomienia systemu jest kombinacja klawiszy **Ctrl+Alt+Del**, jednak w normalnych warunkach korzystanie z niej nie jest zalecane.

2.10. Obsługiwany sprzęt

W obecnej chwili FreeBSD działa na komputerach z magistralami ISA, VLB, EISA i PCI wyposażonych w procesory Intel, AMD, Cyrix lub NexGen „x86”, jak również na komputerach z procesorem Compaq Alpha. Obsługiwane są także dyski IDE i ESDI, rozmaite kontrolery SCSI, karty PCMCIA, urządzenia USB oraz karty sieciowe i szeregowo. FreeBSD pracuje także z szyną microchannel (MCA) firmy IBM.

Lista obsługiwanych urządzeń dołączona jest do każdego wydania FreeBSD w dokumencie FreeBSD Hardware Notes. Można go zwykle znaleźć w pliku `HARDWARE.TXT`, umieszczonym bezpośrednio w głównym katalogu płyty CDROM lub na serwerze FTP, bądź w menu dokumentacji **sysinstall**. Na liście zebrano urządzenia, które poprawnie współpracują z FreeBSD. Kopie tej listy dla różnych wydań systemu i różnych architektur można także znaleźć na podstronie Release Information (<http://www.FreeBSD.org/releases/index.html>) na stronie WWW FreeBSD.

2.11. Rozwiązywanie problemów

W tej części opisujemy, jak radzić sobie z podstawowymi problemami spotykanymi podczas instalacji. W kilku pytaniach i odpowiedziach omawiamy także możliwość uruchamiania FreeBSD i MS-DOS na tym samym komputerze.

2.11.1. Co robić, gdy coś pójdzie nie tak

Ze względu na rozmaite ograniczenia architektury PC, rozpoznawanie urządzeń może niekiedy sprawiać problemy. Można jednak spróbować sobie z nimi poradzić.

Zapoznajmy się z dokumentem Hardware Notes, by mieć pewność, że nasze urządzenia są obsługiwane przez FreeBSD.

Jeśli wciąż występują problemy, mimo, że nasz sprzęt jest obsługiwany, powinniśmy ponownie uruchomić komputer i wybrać opcję wizualnej konfiguracji jądra (visual kernel configuration). Będziemy mieć możliwość przejrzenia naszych urządzeń i podania systemowi informacji o nich. Jądro uruchamiane z dyskietki startowej zakłada, że większość urządzeń skonfigurowanych jest z fabrycznymi ustawieniami IRQ, portów we/wy i kanałów DMA. Jeśli konfiguracja naszego sprzętu jest odmienna, zapewne będziemy musieli poinformować o tym FreeBSD, odpowiednio modyfikując konfigurację.

Może się zdarzyć, że próba rozpoznania urządzenia nieistniejącego spowoduje kłopoty z późniejszym rozpoznawaniem urządzeń rzeczywiście zainstalowanych w komputerze. W takim wypadku powinniśmy wyłączyć sterowniki powodujące konflikty.

Notatka: Pewnych problemów z instalacją można uniknąć dzięki instalacji nowszego oprogramowania sprzętowego (ang. firmware) urządzenia, zwykle płyty głównej. Oprogramowanie sprzętowe płyty głównej znane jest pod nazwą BIOS. Większość producentów płyt głównych lub komputerów umieszcza informacje o nowych wersjach oprogramowania na swoich stronach WWW.

Producenci zwykle stanowczo odradzają instalowanie nowego BIOS-u, oprócz sytuacji, w których jest to uzasadnione, na przykład w przypadku wykrycia poważnego błędu. Instalacja nowszej wersji *może* się nie udać, powodując trwałe uszkodzenie układu BIOS.

Ostrzeżenie Nie należy wyłączać sterowników potrzebnych podczas instalacji, na przykład sterownika ekranu (`sc0`). Jeżeli po zakończeniu konfiguracji jądra instalacja w tajemniczy sposób zastyga lub przerywa pracę, zapewne usunęliśmy lub zmodyfikowaliśmy coś, co nie powinno być ruszane. Musimy ponownie uruchomić komputer i spróbować jeszcze raz.

Podczas konfiguracji możemy:

- Przejrzeć listę sterowników zainstalowanych w jądrze.
- Wyłączyć sterowniki urządzeń, których nie ma w komputerze.
- Zmienić ustawienia IRQ, DRQ i portów we/wy używanych przez sterowniki.

Po dostosowaniu konfiguracji jądra do naszego sprzętu, wpisujemy Q, by ponownie uruchomić komputer z nowymi ustawieniami. Zmiany konfiguracji są trwałe i będą obowiązywać również po zakończeniu instalacji, nie będzie więc trzeba konfigurować jądra na nowo przy każdym uruchamianiu systemu. Jest jednak bardzo prawdopodobne, że będziemy chcieli zbudować niestandardowe jądro.

2.11.2. Jak poradzić sobie z istniejącymi partycjami MS-DOS®

Wielu użytkowników instaluje FreeBSD na komputerach PC z systemem operacyjnym z rodziny Microsoft. Specjalnie dla tych użytkowników przygotowany został program **FIPS**. Narzędzie to znajduje się na płycie instalacyjnej w katalogu `tools`. Można je również pobrać z wielu serwerów lustrzanych FreeBSD.

FIPS umożliwia podzielenie istniejącej partycji MS-DOS na dwie części, zachowując pierwotną partycję i pozwalając na instalację FreeBSD na wolnej drugiej części. Wpierw należy wykonać defragmentację partycji MS-DOS za pomocą dostępnego w Windows narzędzia (w Eksploratorze nacisnąć prawym przyciskiem myszki na dysku twardym, następnie wybrać opcję defragmentacji dysku), albo **Norton Disk Tools**. Następnie należy uruchomić **FIPS**. Program zapyta o potrzebne mu informacje. Potem można ponownie uruchomić komputer i zainstalować FreeBSD na nowym wolnym segmencie. W menu **Distributions** można dowiedzieć się, ile miejsca na dysku będzie w przybliżeniu potrzebne.

Jest także bardzo użyteczny program firmy PowerQuest (<http://www.powerquest.com> (<http://www.powerquest.com/>)), o nazwie **PartitionMagic**. Ma on znacznie większe możliwości niż **FIPS** i stosowanie go jest zalecane, jeśli planuje się częste instalowanie i usuwanie systemów operacyjnych. Nie jest on jednak za darmo; jeśli FreeBSD ma być zainstalowane raz na dobre, **FIPS** zapewne w zupełności wystarczy.

2.11.3. Wykorzystanie systemów plików MS-DOS i Windows®

W chwili obecnej FreeBSD nie obsługuje systemów plików skompresowanych za pomocą programu **Double Space™**. Tym samym musimy wpierw rozkompresować system plików nim FreeBSD będzie mógł odczytać zapisane w nim dane. Można do tego wykorzystać **Agenta kompresji** z menu **Start > Programy > Narzędzia systemowe**.

FreeBSD obsługuje systemy plików MS-DOS. By je zamontować należy wykorzystać polecenie `mount_msdosfs(8)` z odpowiednimi parametrami. Typowa forma polecenia wygląda następująco:

```
# mount_msdosfs /dev/ad0s1 /mnt
```

W tym przykładzie system plików MS-DOS zlokalizowany jest na pierwszej partycji pierwszego dysku twardego. By sprawdzić jak jest w naszym przypadku należy sprawdzić wynik poleceń `dmesg` oraz `mount`. Powinno to pozwolić nam zorientować się w układzie partycji na dysku.

Notatka: Rozszerzone partycje MS-DOS odwzorowywane są na końcu pozostałych „segmentów” we FreeBSD. Przykładowo, pierwsza partycja MS-DOS może znajdować się na `/dev/ad0s1`, partycja FreeBSD na `/dev/ad0s2`, natomiast rozszerzona partycja MS-DOS na `/dev/ad0s3`. Może to być mylące na początku.

Analogicznie można montować partycje NTFS wykorzystując polecenie `mount_ntfs(8)`.

2.11.4. Pytania użytkowników komputerów Alpha

Oto niektóre z najczęściej zadawanych pytań dotyczących instalowania FreeBSD na komputerach Alpha.

1. Czy mogę ładować system z konsoli ARC lub Alpha BIOS?

Nie. FreeBSD, podobnie jak Compaq Tru64 i VMS, może być ładowany tylko z konsoli SRM.

2. Pomocy, brakuje mi miejsca na dysku! Czy muszę wszystko skasować?

Niestety tak.

3. Czy można montować systemy plików Compaq Tru64 lub VMS?

Nie, przynajmniej na razie.

2.12. Instalacja zaawansowana

Napisał Valentino Vaschetto.

W tej części omówiona została instalacja FreeBSD w sytuacjach wyjątkowych.

2.12.1. Instalacja FreeBSD na komputerze bez monitora lub klawiatury

Ten rodzaj instalacji zwany jest „instalacją bez głowy”, ponieważ komputer, na którym FreeBSD będzie instalowane nie ma podłączonego monitora, lub nawet nie ma wyjścia VGA. Jak to możliwe? Dzięki konsoli szeregowej. W roli konsoli szeregowej używa się zwykle innego komputera, który pełni rolę ekranu i klawiatury dla pozbawionego tych urządzeń komputera. By zainstalować system tą metodą, musimy przygotować dyskietki instalacyjne zgodnie z opisem w Sekcja 2.2.7.

By zmodyfikować dyskietki do pracy z konsolą szeregową należy wykonać następujące kroki:

1. Włączenie konsoli szeregowej na dyskietce startowej

Jeśli spróbowałibyśmy uruchomić komputer korzystając z utworzonych właśnie dyskietek startowych, zostałyaby uruchomiona zwykła instalacja FreeBSD. My jednak chcemy, by podczas instalacji używana była konsola szeregowa. By to skonfigurować, montujemy dyskietkę `kern.flp` we FreeBSD przy użyciu polecenia `mount(8)`.

```
# mount /dev/fd0 /mnt
```

Po zamontowaniu dyskietki, wchodzimy do katalogu `/mnt`:

```
# cd /mnt
```

Teraz włączymy na dyskietce konsolę szeregową. Musimy stworzyć plik `boot.config` zawierający wiersz `/boot/loader -h`. Jego zadaniem jest po prostu nakazanie programowi ładującemu system, by używał konsoli szeregowej.

```
# echo "/boot/loader -h" > boot.config
```

Po prawidłowym skonfigurowaniu dyskietki odmontowujemy ją poleceniem `umount(8)`:

```
# cd /
# umount /mnt
```

Możemy wyjąć dyskietkę ze stacji dyskietek.

2. Podłączenie kabla null-modem

Dwa komputery łączymy kablem null-modem. Po prostu podłączamy kabel do portów szeregowych w jednym i drugim komputerze. Zwykły kabel szeregowy nie nadaje się do tego celu, potrzebny jest kabel null-modem, ponieważ jego przewody są odpowiednio skrzyżowane.

3. Uruchomienie instalacji

Możemy już uruchomić instalację. Do stacji dyskiety „bezgłowego” komputera, na którym ma być zainstalowane FreeBSD, wkładamy dyskietkę `kern.flp` i włączamy komputer.

4. Połączenie z „bezgłowym” komputerem

Z komputerem łączymy się korzystając z `cu(1)`:

```
# cu -l /dev/cuaa0
```

Gotowe! Powinniśmy być w stanie kontrolować „bezgłowy” komputer poprzez sesję `cu`. Zostaniemy poproszeni o włożenie dyskietki `mfsroot.flp`, następnie o wybranie typu terminala. Wybieramy kolorową konsolę FreeBSD (FreeBSD color console) i kontynuujemy instalację.

2.13. Przygotowanie własnego nośnika instalacji

Notatka: Dla uproszczenia, w niniejszej części „dysk FreeBSD” oznaczać będzie płytę CDROM lub DVD z FreeBSD, który zakupiliśmy lub przygotowaliśmy samodzielnie.

Może się zdarzyć sytuacja, w której będziemy musieli przygotować własny nośnik lub źródło dla instalacji FreeBSD. Może to być nośnik fizyczny, na przykład taśma, albo inne źródło z którego **sysinstall** będzie mógł pobrać pliki, na przykład lokalny serwer FTP lub partycja MS-DOS.

Oto przykład:

- Mamy wiele komputerów w sieci lokalnej i jeden dysk FreeBSD. Chcemy przygotować lokalny serwer FTP z zawartością dysku FreeBSD, aby komputery mogły z niego korzystać zamiast łączyć się z Internetem.
- Mamy dysk FreeBSD, jednak FreeBSD nie obsługuje naszego napędu CD/DVD. Napęd jest natomiast prawidłowo obsługiwany w MS-DOS/Windows. Chcemy skopiować pliki instalacyjne FreeBSD na partycję DOS i wykorzystać ją do zainstalowania FreeBSD.
- Komputer, na którym chcemy zainstalować system nie ma napędu CD/DVD ani karty sieciowej. Jest inny komputer, który ma napęd CD/DVD lub kartę sieciową i możemy połączyć się z nim kablem szeregowym lub równoległym.
- Chcemy przygotować taśmę, przy pomocy której będzie można zainstalować FreeBSD.

2.13.1. Przygotowanie płyty instalacyjnej

W ramach każdego wydania systemu Projekt FreeBSD udostępnia pięć obrazów płyt CD („obrazów ISO”). Jeśli dysponujemy nagrywarką CD, możemy je nagrać („wypalić”) na płytach, otrzymując zestaw płyt, które mogą posłużyć do zainstalowania systemu. Jest to najprostszy sposób instalacji FreeBSD w przypadku, gdy mamy nagrywarkę i tanie połączenie z Internetem.

1. Pobranie obrazów ISO

Obrazy ISO każdego z wydań systemu można pobrać z

`ftp://ftp.FreeBSD.org/pub/FreeBSD/ISO-IMAGES-arch/version` lub z najbliższego serwera lustrzanego. W miejscu *arch* i *version* wstawiamy odpowiednią nazwę architektury i wersję.

Wspomniany katalog zawiera zwykle następujące obrazy:

Tabela 2-5. Nazwy obrazów ISO dla FreeBSD 4.x i ich znaczenie

Nazwa pliku	Zawartość
<code>version-RELEASE-arch-miniinst.iso</code>	Wszystko, co jest potrzebne do zainstalowania FreeBSD.
<code>version-RELEASE-arch-disc1.iso</code>	Wszystko, co jest potrzebne do zainstalowania FreeBSD, i tyle dodatkowych pakietów, ile zmieściło się na płycie.
<code>version-RELEASE-arch-disc2.iso</code>	„Żywy system plików”, używany wraz z dostępną w sysinstall funkcją „Repair” (naprawa). Kopia drzewa CVS FreeBSD. Dodatkowe pakiety o charakterze niezależnym.

Tabela 2-6. Nazwy obrazów ISO dla FreeBSD 5.x i ich znaczenie

Nazwa pliku	Zawartość
<code>version-RELEASE-arch-bootonly.iso</code>	Wszystko co jest niezbędne by uruchomić jądro FreeBSD i rozpocząć instalację. Pliki instalacyjne zostaną pobrane z serwera FTP bądź innego źródła.
<code>version-RELEASE-arch-miniinst.iso</code>	Wszystko, co jest potrzebne do zainstalowania FreeBSD.
<code>version-RELEASE-arch-disc1.iso</code>	Wszystko co jest potrzebne by zainstalować FreeBSD jako „żywy system plików” używany wraz z dostępną w sysinstall funkcją „Repair” (naprawa).
<code>version-RELEASE-arch-disc2.iso</code>	Dokumentacja FreeBSD i tyle dodatkowych pakietów, ile zmieściło się na płycie.

Musimy pobrać albo obraz ISO mini, albo obraz pierwszej płyty. Nie ma sensu pobierać obydwu, ponieważ obraz pierwszej płyty zawiera wszystko to, co obraz mini.

Notatka: Obraz ISO mini dostępny jest tylko dla wydań starszych niż FreeBSD 5.4-RELEASE.

Z obrazu ISO miniinst warto jest skorzystać, gdy mamy niedrogi dostęp do Internetu. Za jego pomocą możemy zainstalować FreeBSD, natomiast niezależne oprogramowanie instalujemy przez Internet, przy pomocy systemu portów i pakietów (patrz: Rozdział 4).

Płytę pierwszą wybieramy wtedy, gdy oprócz zainstalowania systemu chcemy skorzystać z zestawu wybranych pakietów oprogramowania.

Pozostałe płyty są przydatne, lecz nie niezbędne, szczególnie, gdy dysponujemy szybkim dostępem do Internetu.

2. Nagranie płyt CD

Pliki obrazów należy nagrać na płyty. Jeśli zamierzamy robić to w systemie FreeBSD, informacje na ten temat znajdziemy w Sekcja 17.6 (w szczególności Sekcja 17.6.3 oraz Sekcja 17.6.4).

Jeżeli płyty nagrywać będziemy w innym systemie, do tego celu możemy posłużyć się dowolnymi dostępnymi programami obsługującymi nagrywkę płyt CD. ISO jest standardowym formatem obrazu płyt obsługiwany w wielu aplikacjach nagrywających.

Notatka: Zainteresowanych przygotowaniem własnych wydań FreeBSD odsyłamy do artykułu Release Engineering (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/releeng)(ang.).

2.13.2. Przygotowanie lokalnego serwera FTP z dyskiem FreeBSD

Układ plików na dysku FreeBSD jest taki sam, jak układ plików na serwerze FTP. Dzięki temu łatwo możemy przygotować lokalny serwer FTP, który może być wykorzystany przez inne komputery w sieci do instalacji FreeBSD.

1. Na komputerze, który będzie służyć jako serwer FTP, umieszczamy CDROM w napędzie i montujemy go w katalogu `/cdrom`.

```
# mount /cdrom
```

2. Zakładamy konto dla anonimowego użytkownika FTP w `/etc/passwd`. Plik `/etc/passwd` modyfikujemy przy użyciu `vipw(8)`. Dodajemy następujący wiersz:

```
ftp:*:99:99::0:0:FTP:/cdrom:/nonexistent
```

3. Na koniec upewniamy się, że usługa FTP jest włączona w `/etc/inetd.conf`.

Od tej chwili każdy, kto jest w stanie nawiązać połączenie z naszym komputerem, może podczas instalacji FreeBSD wybrać jako źródło serwer FTP, w menu wyboru serwera FTP wybrać opcję „Other” (inny) i wpisać

```
ftp://nasz.komputer.
```

Notatka: Jeśli nośnik, z którego uruchamiamy instalator (najczęściej dyskietka), nie pochodzi z dokładnie tej samej wersji co pliki na naszym serwerze FTP, to **sysinstall** nie pozwoli nam kontynuować instalacji. By pominąć tę blokadę należy w menu Options zmienić nazwę dystrybucji na any.

Ostrzeżenie Ta metoda może być z powodzeniem stosowana na komputerze w sieci lokalnej, chronionym przez zaporę ogniową. Udostępnianie serwera FTP innym użytkownikom Internetu (a nie tylko sieci lokalnej) naraża nasz komputer na ataki włamywaczy i inne problemy. Decydując się na to należy koniecznie przestrzegać zasad bezpieczeństwa.

2.13.3. Przygotowywanie dyskietek instalacyjnych

Jeżeli koniecznie chcemy instalować system z dyskietek (co *nie jest* zalecane), na przykład z powodu nieobsługiwania urządzenia lub po prostu z zamiłowania do utrudnień, musimy najpierw przygotować dyskietki instalacyjne.

Będziemy potrzebować co najmniej tylu dyskietek 1.44 MB lub 1.2 MB, by zmieściły się na nich wszystkie pliki z katalogu `bin` (binarne pliki dystrybucyjne). Jeśli dyskietki przygotowujemy w DOS-ie, to *muszą* one być sformatowane przy pomocy DOS-owego polecenia `FORMAT`. W Windows do sformatowania dyskietek możemy użyć Explorera (klikamy prawym przyciskiem myszy na stacji A: i wybieramy „Format”).

Nie ufajmy dyskietkom sformatowanym fabrycznie. Dla pewności sformatujmy je jeszcze raz samodzielnie. W przeszłości wiele problemów zgłaszanych przez użytkowników spowodowanych było korzystaniem z nieprawidłowo sformatowanych dyskietek, dlatego też zwracamy na to uwagę.

Jeżeli do przygotowania dyskietek służy nam komputer z FreeBSD, również powinniśmy je sformatować. Dyskietki nie muszą być formatowane w DOS-owym systemie plików. Możemy utworzyć na nich system plików UFS, za pomocą poleceń `bsdlabel` i `newfs`, wywołanych w następujący sposób (na przykładzie dyskietek 3.5" 1.44 MB):

```
# fdformat -f 1440 fd0.1440
# bsdlabel -w -r fd0.1440 floppy3
# newfs -t 2 -u 18 -l 1 -i 65536 /dev/fd0
```

Notatka: W przypadku dyskietek 5.25" 1.2 MB, wpisalibyśmy odpowiednio `fd0.1200` i `floppy5`.

Po takiej operacji dyskietki będzie można zamontować i zapisywać na nich dane tak samo, jak na innych systemach plików.

Po sformatowaniu dyskietek należy skopiować na nie pliki. Pliki dystrybucyjne podzielone są na kawałki o wygodnych rozmiarach, tak aby pięć z nich mieściło się na typowej dyskietce 1.44 MB. Umieścimy na każdej z dyskietek tyle plików, ile się zmieści, aż wszystkie pliki dystrybucyjne znajdą się na dyskietkach. Pliki powinny być umieszczone w odpowiednim katalogu na dyskietce, np.: `a:\bin\bin.aa`, `a:\bin\bin.ab`, itd.

Podczas instalacji, gdy pojawi się ekran wyboru nośnika (Media), wybieramy Floppy (dyskietki). Dalej poprowadzi nas program instalacyjny.

2.13.4. Instalacja z partycji MS-DOS

By można było zainstalować FreeBSD z partycji MS-DOS, kopiujemy pliki dystrybucyjne do katalogu `freebsd` w głównym katalogu partycji - na przykład `c:\freebsd`. Wewnątrz tego katalogu musi być częściowo zachowana struktura katalogów płyty CDROM lub serwera FTP, jeśli więc kopiujemy pliki z płyty CD, dobrze jest skorzystać z DOS-owego polecenia `xcopy`. Dla przykładu, poniższe polecenia przygotowują minimalną instalację FreeBSD:

```
C:\> md c:\freebsd
C:\> xcopy e:\bin c:\freebsd\bin\ /s
C:\> xcopy e:\manpages c:\freebsd\manpages\ /s
```

W przykładzie założyliśmy, że miejsce dla FreeBSD mamy na dysku `C:`, a napęd CDROM dostępny jest jako dysk `E:`.

Jeśli nie dysponujemy napędem CDROM, pliki dystrybucyjne możemy pobrać z ftp.FreeBSD.org ([ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/9.1-RELEASE/](http://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/9.1-RELEASE/)). Każdy zestaw plików umieszczony jest w oddzielnym katalogu; na przykład zestaw *base* znajduje się w katalogu `9.1/base/` ([ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/9.1-RELEASE/base/](http://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/9.1-RELEASE/base/)).

Zestawy plików, które chcemy instalować z partycji MS-DOS (i dla których jest na niej odpowiednio dużo wolnego miejsca), umieszczamy w katalogu `c:\freebsd`. Na potrzeby instalacji minimalnej wystarczy zestaw `BIN`.

2.13.5. Przygotowanie taśmy instalacyjnej

Instalacja z taśmy jest jedną z najprostszych metod, obok instalacji przez FTP i instalacji z płyty CD. Program instalacyjny zakłada, że taśma po prostu zawiera pliki w postaci archiwum tar. Interesujące nas pliki dystrybucyjne archiwizujemy na taśmie:

```
# cd /freebsd/distdir
# tar cvf /dev/rwt0 dist1 ... dist2
```

Przeprowadzając instalację powinniśmy upewnić się, że dysponujemy odpowiednią ilością wolnego miejsca w jakimś katalogu tymczasowym (będziemy mieć możliwość wyboru tego katalogu), by pomieścić *pełną* zawartość przygotowanej wcześniej taśmy. Ze względu na to, że dostęp do danych na taśmie nie jest swobodny, taki rodzaj instalacji będzie wymagać dość sporej przestrzeni tymczasowej. Można założyć, że potrzeba będzie tyle przestrzeni, ile zajmują dane zapisane na taśmie.

Notatka: Rozpoczynając instalację pamiętajmy, by taśma była umieszczona w napędzie *przed* uruchomieniem komputera z dyskietki startowej. W przeciwnym razie napęd taśmowy może nie zostać wykryty podczas rozpoznawania urządzeń.

2.13.6. Przed instalacją przez sieć

Są trzy możliwości instalacji przez sieć: port szeregowy (SLIP lub PPP), port równoległy (PLIP (kabel laplink)) lub Ethernet (typowa karta sieciowa Ethernet (także PCMCIA)).

Obsługa protokołu SLIP jest dosyć prymitywna i ogranicza się do bezpośrednich połączeń, jak choćby kabel łączący komputer przenośny z innym komputerem. Połączenie musi być bezpośrednie, ponieważ instalacja za pośrednictwem SLIP nie umożliwia dzwonienia; jest to możliwe w przypadku PPP, dlatego też powinno się używać PPP zamiast SLIP, o ile to możliwe.

Jeżeli korzystamy z modemu, to PPP jest najprawdopodobniej jedyną możliwością. Zawczasu przygotujmy sobie informacje od dostawcy usług sieciowych, ponieważ będą nam one potrzebne na wczesnym etapie instalacji.

Jeśli łącząc się z dostawcą usług sieciowych używamy PAP lub CHAP (innymi słowy, jeśli w Windows możemy uzyskać połączenie bez korzystania ze skryptu), wówczas wystarczy, że w linii poleceń **ppp** wpisujemy **dia1**. W przeciwnym razie będziemy musieli połączyć się z dostawcą usług sieciowych za pomocą „poleceń AT”, zależnych od typu modemu, gdyż do dyspozycji będziemy mieć jedynie uproszczony emulator terminala. Więcej informacji znajdziemy w poświęconych user-ppp częściach Podręcznika i FAQ (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/books/faq/ppp.html). Jeśli wystąpią problemy, możemy posłużyć się poleceniem `set log local ...`, by komunikaty były pokazywane na ekranie.

Jeżeli dysponujemy bezpośrednim połączeniem z innym komputerem z FreeBSD (w wersji 2.0-R lub późniejszej), wówczas mamy również możliwość instalacji przez port równoległy. Prędkość transmisji danych portem równoległym jest zwykle znacznie wyższa niż prędkość przesyłania portem szeregowym (do 50 kilobajtów/sekundę), dzięki czemu instalacja przebiega szybciej.

Najszybszym wariantem instalacji poprzez sieć jest wykorzystanie karty sieciowej Ethernet. FreeBSD obsługuje większość popularnych kart sieciowych; lista obsługiwanych kart (wraz z ich ustawieniami) znajduje się w dokumencie Hardware Notes, dołączonym do każdego wydania FreeBSD. Jeżeli korzystamy z karty sieciowej PCMCIA, pamiętajmy o tym, by była ona włożona *przed* włączeniem komputera. Niestety, jak dotąd FreeBSD nie obsługuje wkładania kart PCMCIA w trakcie instalacji.

Będziemy musieli znać nasz adres IP, maskę podsieci, oraz nazwę naszego komputera. Jeśli instalujemy za pośrednictwem PPP i nie mamy statycznego adresu IP, nie musimy się przejmować, gdyż adres IP może być przydzielony dynamicznie przez dostawcę usług. Administrator sieci może nam odpowiedzieć, jakie parametry podać podczas konfiguracji sieci. Jeśli do połączeń z innymi stacjami będziemy używać ich nazw, a nie adresów IP, to dodatkowo będziemy musieli znać adres serwera nazw i prawdopodobnie adres bramy (w przypadku PPP jest to adres IP dostawcy). Jeżeli mamy zamiar instalować za pośrednictwem FTP i proxy HTTP, będzie nam ponadto potrzebny adres proxy. Skontaktujmy się z administratorem sieci lub dostawcą usług sieciowych *przed* rozpoczęciem instalacji, jeśli nie znamy któregoś z wymienionych powyżej adresów.

2.13.6.1. Przed instalacją przez NFS

Instalacja przez NFS jest raczej mało skomplikowana. Wystarczy po prostu skopiować wybrane pliki dystrybucyjne na serwer, następnie podczas instalacji wybrać NFS jako nośnik i wskazać serwer.

Jeżeli serwer wymaga stosowania „uprzywilejowanego portu” (zwykle jest tak w przypadku stacji roboczych Sun), musimy to zaznaczyć w menu **Options** (opcja `NFS Secure`), zanim rozpoczniemy instalację.

Jeśli nasza karta sieciowa jest niezbyt dobrej jakości i nie grzeszy prędkością, możemy włączyć opcję `NFS Slow`.

Instalacja przez NFS wymaga, by serwer obsługiwał montowanie podkatalogów, na przykład jeśli katalog dystrybucyjny FreeBSD 9.1 znajduje się w: `ziggy:/usr/archive/stuff/FreeBSD`, to serwer ziggy musi umożliwiać bezpośrednie montowanie katalogu `/usr/archive/stuff/FreeBSD`, a nie tylko `/usr`, lub `/usr/archive/stuff`.

We FreeBSD w pliku `/etc/exports` możliwość montowania podkatalogów włącza się opcją `-alldirs`. W innych serwerach NFS może być inaczej. Jeśli otrzymujemy od serwera komunikaty o treści „permission denied” (odmowa dostępu), prawdopodobnie jest to spowodowane właśnie nieprawidłowym ustawieniem wspomnianej opcji.

Rozdział 3. Podstawy Uniksa

Rozdział na nowo napisał Chris Shumway. Tłumaczył Michał Wojciechowski.

3.1. Streszczenie

W niniejszym rozdziale omówione zostaną podstawowe polecenia i możliwości systemu operacyjnego FreeBSD. Wiele informacji dotyczyć będzie ogółem systemów typu UNIX. Czytelnikom zaznajomionym z tą tematyką w zupełności wystarczy pobieżne przejrzanie rozdziału. Natomiast ci, którzy dopiero rozpoczynają swoją przygodę z FreeBSD, powinni przeczytać go bardzo uważnie.

Po przeczytaniu tego rozdziału będziemy wiedzieć:

- Jak korzystać z „konsol wirtualnych” FreeBSD.
- Jak działają prawa dostępu do plików i flagi plików we FreeBSD.
- Jaki jest domyślny układ systemu plików FreeBSD.
- Jaka jest organizacja dysku we FreeBSD.
- Jak montować i odmontowywać systemy plików.
- Czym są procesy, demony i sygnały.
- Co to jest powłoka, oraz jak można zmienić własne środowisko pracy.
- Jak posługiwać się prostymi edytorami tekstu.
- Jaki jest związek pomiędzy urządzeniami i plikami węzłowymi urządzeń.
- Jaki format binarny jest wykorzystywany we FreeBSD.
- W jaki sposób korzystać z dokumentacji systemowej w poszukiwaniu dodatkowych informacji.

3.2. Konsole wirtualne i terminale

Z systemu FreeBSD korzystać można na różne sposoby; jednym z nich jest wpisywanie poleceń w terminalu tekstowym. Większość systemów operacyjnych typu UNIX dostępna jest właśnie poprzez polecenia. W niniejszej części dowiemy się, czym są „terminale” i „konsole”, oraz jak się nimi posługiwać we FreeBSD.

3.2.1. Konsola

Jeśli konfigurując FreeBSD nie wybraliśmy, by przy uruchamianiu systemu było automatycznie ładowane środowisko graficzne, to po uruchomieniu i wykonaniu skryptów startowych system przywita nas komunikatem logowania się do systemu. Zobaczmy mniej więcej coś takiego:

```
Additional ABI support:.  
Local package initialization:.  
Additional TCP options:.
```

```
Fri Sep 20 13:01:06 EEST 2002
```

```
FreeBSD/i386 (pc3.example.org) (ttyv0)
```

```
login:
```

Na różnych komputerach komunikat ten może wyglądać nieco inaczej, jednak z pewnością będzie podobny. W tej chwili interesują nas jego dwa ostatnie wiersze. Wiersz drugi od końca ma postać:

```
FreeBSD/i386 (pc3.example.org) (ttyv0)
```

Widać tu kilka informacji o systemie, który właśnie został uruchomiony. Mamy przed oczami konsolę „FreeBSD”, działającą na komputerze z procesorem firmy Intel (lub kompatybilnym) z rodziny x86¹. Komputer ten został nazwany (każdy komputer uniksowy ma nazwę) `pc3.example.org` i w tej chwili widoczna jest jego konsola systemowa — terminal `ttyv0`.

Ostatni wiersz ma zawsze taką postać:

```
login:
```

Tu wpisujemy „nazwę użytkownika”, by zalogować się do systemu. Opis tej czynności przedstawiony jest w kolejnej części.

3.2.2. Logowanie się do FreeBSD

FreeBSD jest systemem wieloużytkownikowym i wielozadaniowym. Tak oficjalnie określa się system, z którego na jednym komputerze może korzystać wiele różnych osób, uruchamiając jednocześnie wiele programów.

Każdy system wieloużytkownikowy musi mieć możliwość odróżnienia jednego „użytkownika” od pozostałych. FreeBSD (i wszystkie systemy uniksopodobne) wymaga, aby użytkownik „zalogował się” do systemu, zanim będzie mógł uruchamiać programy. Każdy użytkownik ma niepowtarzalną nazwę („nazwę użytkownika”) oraz sobie tylko znany klucz („hasło”). FreeBSD wymaga wpisania jednego i drugiego, zanim zezwoli użytkownikowi na uruchamianie jakichkolwiek programów.

Zaraz po załadowaniu systemu i zakończeniu uruchamiania skryptów startowych², FreeBSD wyświetli komunikat z prośbą o podanie nazwy użytkownika:

```
login:
```

Dla przykładu założmy, że nasz użytkownik nazywa się `janeek`. Wpisujemy tutaj `janeek` i naciskamy **Enter**. Powinniśmy zostać poproszeni o podanie „hasła”:

```
login: janeek
Password:
```

Następnie wpisujemy hasło `janka`, i naciskamy **Enter**. Hasło *nie pojawia się!* Na razie nie będziemy się tym zajmować. Wystarczy wiedzieć, że dzieje się tak ze względów bezpieczeństwa.

Jeśli podaliśmy prawidłowe hasło, powinniśmy być już zalogowani do FreeBSD, i gotowi do eksperymentowania z dostępnymi poleceniami.

Powinniśmy zobaczyć wiadomość dnia (ang. message of the day MOTD) oraz znak zachęty (`#`, `$` bądź `%`). Oznacza to, że udało nam się zalogować do FreeBSD.

3.2.3. Konsole wirtualne

Polecenia uniksowe można z powodzeniem wpisywać na jednej konsoli, jednak FreeBSD potrafi wykonywać wiele programów jednocześnie. Korzystanie z jednej konsoli do wydawania poleceń zakrawa na marnotrawstwo, ponieważ system zdolny jest obsłużyć w jednej chwili całe mnóstwo programów. W wykorzystaniu tej możliwości bardzo pomocne są „konsole wirtualne”.

Konfigurując FreeBSD możemy uaktywnić wiele konsol wirtualnych. Z dowolnej z nich możemy się przełączyć na inną naciskając odpowiednią kombinację klawiszy. Każda konsola ma własny kanał wyjściowy, FreeBSD zajmuje się odpowiednim przekazywaniem informacji wprowadzanych z klawiatury i wypisywanych na ekranie, gdy dochodzi do przełączenia konsoli na inną.

Pewne kombinacje klawiszy używane są do przechodzenia między konsolami³. Kombinacje **Alt-F1**, **Alt-F2**, aż do **Alt-F8** służą do przełączania na kolejną konsolę wirtualną.

Przechodząc z jednej konsoli na inną, FreeBSD zajmuje się zachowaniem i odtworzeniem wyglądu ekranu. W efekcie otrzymujemy „złudzenie” posiadania wielu „wirtualnych” ekranów i klawiatur, które mogą służyć do wydawania poleceń systemowi FreeBSD. Programy uruchomione na jednej z konsol nie przerywają swej pracy, gdy ta konsola przestaje być widoczna — po przejściu na inną konsolę wirtualną programy kontynuują swoje działanie.

3.2.4. Plik `/etc/ttys`

Zgodnie z domyślną konfiguracją FreeBSD uruchamia osiem konsol wirtualnych. Nie jest to jednak permanentne ustawienie, i może być w łatwy sposób zmienione, aby konsol wirtualnych było więcej lub mniej. Plik `/etc/ttys` odpowiedzialny jest za liczbę konsol wirtualnych i ich konfigurację.

Modyfikując plik `/etc/ttys` możemy zmieniać konfigurację konsol wirtualnych FreeBSD. Każdy nie będący komentarzem wiersz tego pliku (czyli wiersz nie rozpoczynający się znakiem `#`) zawiera ustawienia jednego z terminali lub konsoli wirtualnej. W domyślnej wersji tego pliku występującej we FreeBSD skonfigurowanych jest 9 konsol wirtualnych, przy czym 8 z nich jest włączonych. Za ich konfigurację odpowiadają wiersze rozpoczynające się symbolem `ttyv`:

#	name	getty	type	status	comments
#	ttyv0	"/usr/libexec/getty Pc"	cons25	on	secure
#	Virtual terminals				
	ttyv1	"/usr/libexec/getty Pc"	cons25	on	secure
	ttyv2	"/usr/libexec/getty Pc"	cons25	on	secure
	ttyv3	"/usr/libexec/getty Pc"	cons25	on	secure
	ttyv4	"/usr/libexec/getty Pc"	cons25	on	secure
	ttyv5	"/usr/libexec/getty Pc"	cons25	on	secure
	ttyv6	"/usr/libexec/getty Pc"	cons25	on	secure
	ttyv7	"/usr/libexec/getty Pc"	cons25	on	secure
	ttyv8	"/usr/X11R6/bin/xdm -nodaemon"	xterm	off	secure

Dokładny opis poszczególnych kolumn tego pliku i opcji, za pomocą których konfiguruje się konsole wirtualne, znaleźć można w dokumentacji systemowej `ttys(5)`.

3.2.5. Konsola trybu jednego użytkownika

„Tryb jednego użytkownika” szczegółowo opisuje Sekcja 12.6.2. Istotne jest, że w trybie jednego użytkownika dostępna jest tylko jedna konsola. Nie jest możliwe korzystanie z konsol wirtualnych. Konfiguracja konsoli trybu jednego użytkownika również znajduje się w pliku `/etc/ttys`. Odpowiada jej wiersz rozpoczynający się słowem `console`:

```
# name  getty                                type    status    comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                                unknown off secure
```

Notatka: Zgodnie z informacją zawartą w komentarzu nad wierszem `console`, wiersz ten można zmodyfikować, zmieniając parametr `secure` na `insecure`. Jeśli tak zrobimy, FreeBSD po uruchomieniu w trybie jednego użytkownika będzie pytał o hasło użytkownika `root`.

Zachowajmy jednak ostrożność, jeśli wpisujemy tu `insecure`. Jeżeli zdarzy się nam zapomnieć hasła użytkownika `root`, może okazać się potrzebne uruchomienie trybu jednego użytkownika. Będzie to nadal możliwe, może jednak być nieco trudne dla osób nie orientujących się w procesie uruchamiania FreeBSD i uczestniczących w nim programach.

3.3. Prawa dostępu

FreeBSD, będąc bezpośrednim potomkiem systemu UNIX BSD, oparte jest na kilku kluczowych założeniach Uniksa. Najbardziej widocznym z nich jest fakt, że FreeBSD jest systemem wieloużytkownikowym — potrafi jednocześnie obsługiwać wielu użytkowników pracujących niezależnie od siebie. System jest odpowiedzialny za właściwe zarządzanie odwołaniami do sprzętu, pamięci i czasu procesora, po równo dla każdego z użytkowników.

Ze względu na obsługę wielu użytkowników, zasoby, którymi zarządza system, mają przypisane prawa dostępu określające, kto może czytać, zapisywać i uruchamiać dany zasób. Prawa dostępu przechowywane są w postaci dwóch oktetów podzielonych na trzy części, z których pierwsza odnosi się do właściciela pliku, druga do grupy posiadającej plik, a trzecia do innych użytkowników. W postaci numerycznej zapisuje się to następująco:

Wartość	Uprawnienia	Symbol
0	Odczyt: nie, zapis: nie, wykonywanie: nie	---
1	Odczyt: nie, zapis: nie, wykonywanie: tak	--x
2	Odczyt: nie, zapis: tak, wykonywanie: nie	-w-
3	Odczyt: nie, zapis: tak, wykonywanie: tak	-wx
4	Odczyt: tak, zapis: nie, wykonywanie: nie	r--

Wartość	Uprawnienia	Symbol
5	Odczyt: tak, zapis: nie, wykonywanie: nie tak	<code>r-x</code>
6	Odczyt: tak, zapis: tak, wykonywanie: nie	<code>rw-</code>
7	Odczyt: tak, zapis: tak, wykonywanie: tak	<code>rwX</code>

Korzystając z polecenia `ls(1)` możemy posłużyć się opcją `-l`, by zawartość katalogu została pokazana w formie szczegółowej, z uwzględnieniem kolumny zawierającej informację o prawach dostępu do pliku dla jego właściciela, grupy, oraz wszystkich innych. Przykładowy wynik polecenia `ls -l`:

```
% ls -l
total 530
-rw-r--r-- 1 root wheel 512 Sep  5 12:31 myfile
-rw-r--r-- 1 root wheel 512 Sep  5 12:31 otherfile
-rw-r--r-- 1 root wheel 7680 Sep  5 12:31 email.txt
...
```

Pierwsza kolumna listy plików po wykonaniu polecenia `ls -l` ma następującą postać:

```
-rw-r--r--
```

Pierwszy znak (od lewej) określa, czy plik jest zwykłym plikiem, katalogiem, urządzeniem znakowym, gniazdem, czy jakimkolwiek innym urządzeniem pseudo-plikowym. Widoczny w przykładzie znak `-` oznacza zwykły plik. Kolejne trzy znaki, w przykładzie są to `rw-`, reprezentują prawa dostępu, którymi dysponuje właściciel pliku. Następne trzy znaki `r--`, określają prawa dostępu grupy, do której należy plik. Ostatnia trójka `r--`, oznacza prawa dostępu dla innych. Minus oznacza brak jednego z praw dostępu. Plik przedstawiony w przykładzie może być więc odczytywany i zapisywany przez swojego właściciela, oraz jedynie odczytywany przez grupę i innych. Zgodnie z powyższą tabelą, prawa dostępu do tego pliku mają wartość `644`, przy czym każda cyfra reprezentuje trzy części uprawnień.

W porządku, ale w jaki sposób system kontroluje dostęp do urządzeń? Zasadniczo większość urządzeń jest traktowana przez FreeBSD jak pliki, które mogą być otwierane, odczytywane i zapisywane podobnie jak wszystkie inne pliki. Specjalne pliki urządzeń przechowywane są w katalogu `/dev`.

Również katalogi traktowane są jak pliki — też są im przypisywane prawa odczytu, zapisu i wykonania. Bit wykonania katalogu ma nieco inne znaczenie niż w przypadku pliku. Posiadanie prawa wykonania katalogu oznacza, że można do niego wejść, czyli posłużyć się poleceniem „`cd`”. Ponadto umożliwia to dostęp do zawartych w katalogu plików o znanych nazwach (oczywiście obowiązują także indywidualne prawa dostępu do każdego z plików).

W szczególności, wyświetlenie listy plików katalogu wymaga posiadania prawa do jego odczytu, natomiast do usunięcia pliku o znanej nazwie potrzebne będą prawa do zapisu i wykonania dla katalogu, w którym ów plik się znajduje.

Jest jeszcze kilka innych bitów uprawnień, jednak są one stosowane w specjalnych przypadkach, np. do włączenia atrybutu SUID, lub „lepkiego” bitu dla katalogu. Więcej informacji o prawach dostępu i o ich przydzielaniu można znaleźć w dokumentacji systemowej polecenia `chmod(1)`.

3.3.1. Uprawnienia symboliczne

Napisał Tom Rhodes. Tłumaczył Cezary Morga.

Uprawnienia symboliczne, określane również jako wyrażenia symboliczne, przy określaniu praw dostępu do plików lub katalogów wykorzystują litery w miejsce wartości liczbowych. Wyrażenia symboliczne wykorzystują składnię: (kto) (akcja) (uprawnienia), przy czym dostępne są następujące wartości:

Opcja	Litera	Znaczenie
(kto)	u	Użytkownik (właściciel)
(kto)	g	Grupa
(kto)	o	Inni
(kto)	a	Wszyscy („świat”)
(akcja)	+	Dodanie uprawnień
(akcja)	-	Usunięcie uprawnień
(akcja)	=	Ustawienie uprawnień
(uprawnienia)	r	Odczyt
(uprawnienia)	w	Zapis
(uprawnienia)	x	Wykonywanie
(uprawnienia)	t	Bit „lepki”
(uprawnienia)	s	Ustawienie UID lub GID

Do ustawienia tych wartości, podobnie jak w przypadku wartości liczbowych, wykorzystywane jest polecenie `chmod(1)`. Przykładowo, by zablokować dostęp innych użytkowników do *PLIKU* należy wpisać:

```
% chmod go= PLIK
```

Gdy musimy wykonać więcej niż jedną zmianę uprawnień parametry należy oddzielić przecinkami. Na przykład, poniższe polecenie usunie prawa zapisu do *PLIKU* grupie i innym. Następnie doda wszystkim prawo wykonywania:

```
% chmod go-w,a+x PLIK
```

3.3.2. Flagi plików we FreeBSD

Napisał Tom Rhodes. Tłumaczył Cezary Morga.

Dodatkowo, oprócz opisanych wyżej praw dostępu, FreeBSD wykorzystuje również „flagi plików”. Flagi te umożliwiają wprowadzenie dodatkowego poziomu ochrony i kontroli plików. Nie dotyczą natomiast katalogów.

Dzięki zwiększonemu poziomowi kontroli plików system może zagwarantować, że w niektórych sytuacjach nawet użytkownik `root` nie będzie mógł usunąć bądź zmodyfikować plików.

Zmiany flag plików dokonuje się poleceniem `chflags(1)`. Przykładowo, by plikowi `plik1` nadać flagę nieusuwalności należy wydać poniższe polecenie:

```
# chflags sunlink plik1
```

Natomiast, by usunąć flagę nieusuwalności wystarczy wprowadzić takie samo polecenie dodając „no” przed `sunlink`:

```
# chflags nosunlink plik1
```

By wyświetlić flagi danego pliku wystarczy wpisać polecenie `ls(1)` z parametrem `-lo`:

```
# ls -lo plik1
```

Wynik powinien być zbliżony do poniższego:

```
-rw-r--r--  1 trhodes  trhodes  sunlnk 0 Mar  1 05:54 plik1
```

Niektóre z flag mogą być dodawane i usuwane jedynie przez użytkownika `root`, podczas gdy inne mogą być ustawiane również przez właściciela pliku. Zaleca się aby administratorzy przeczytali strony podręcznika systemowego `chflags(1)` oraz `chflags(2)`.

3.4. Struktura katalogów

Poznanie hierarchii katalogów FreeBSD jest podstawą ogólnego zrozumienia działania systemu. Najważniejszym zagadnieniem jest koncepcja katalogu głównego, „/”. Jest on montowany jako pierwszy podczas uruchamiania systemu i zawiera podstawowe pliki niezbędne do przygotowania systemu do pracy w trybie wieloużytkownikowym. Ponadto w katalogu głównym znajdują się punkty montowania innych systemów plików, które możemy montować.

Punktem montowania nazywany jest katalog, poprzez który inny system plików może być dołączony do głównego systemu plików. Sekcja 3.5 zawiera więcej informacji. Przykładem typowego punktu montowania może być `/usr`, `/var`, `/tmp`, `/mnt` oraz `/cdrom`. Najczęściej każdemu z takich katalogów odpowiada wpis w pliku `/etc/fstab`. Plik ten zawiera tabelę systemów plików i ich punktów montowania, z której korzysta system. Większość systemów plików wymienionych w `/etc/fstab` jest montowana automatycznie przez skrypt `rc(8)` podczas uruchamiania systemu, wyjątkiem są te wpisy, które mają opcję `noauto`. Sekcja 3.6.1 zawiera więcej informacji.

Pełny opis struktury systemu plików znajduje się w dokumentacji systemowej `hier(7)`. Tu ograniczymy się do pobieżnego zapoznania się z najważniejszymi katalogami.

Katalog	Opis
/	Główny katalog systemu plików.
/bin/	Programy użytkowe wykorzystywane zarówno w trybie jednego użytkownika, jak i w trybie wielu użytkowników.
/boot/	Programy i pliki konfiguracyjne używane podczas uruchamiania systemu.
/boot/defaults/	Pliki z domyślną konfiguracją uruchamiania systemu; patrz <code>loader.conf(5)</code> .
/dev/	Pliki urządzeń; patrz <code>intro(4)</code> .
/etc/	Pliki i skrypty konfiguracyjne.
/etc/defaults/	Pliki z domyślną konfiguracją systemu; patrz <code>rc(8)</code> .
/etc/mail/	Pliki konfiguracyjne dla serwerów poczty, na przykład <code>sendmail(8)</code> .
/etc/namedb/	Pliki konfiguracyjne programu <code>named</code> ; patrz <code>named(8)</code> .

Katalog`/etc/periodic/``/etc/ppp/``/mnt/``/proc/``/rescue/``/root/``/sbin/``/stand/``/tmp/``/usr/``/usr/bin/``/usr/include/``/usr/lib/``/usr/libdata/``/usr/libexec/``/usr/local/``/usr/obj/``/usr/ports``/usr/sbin/``/usr/share/`**Opis**

Skrypty uruchamiane raz dziennie, raz na tydzień i raz na miesiąc za pośrednictwem `cron(8)`; patrz `periodic(8)`.

Pliki konfiguracyjne `ppp`; patrz `ppp(8)`.

Pusty katalog, najczęściej wykorzystywany przez administratorów jako tymczasowy punkt montowania..

System plików procesów, patrz `procfs(5)`, `mount_procfs(8)`.

Katalog zawierający programy przydatne w przypadku awarii; patrz `rescue(8)`.

Katalog domowy użytkownika `root`.

Programy i narzędzia administracyjne wykorzystywane zarówno w trybie jednego użytkownika, jak i w trybie wielu użytkowników.

Programy używane w samodzielnym środowisku.

Pliki tymczasowe. Zawartość katalogu `/tmp` NIE JEST zachowywana przy ponownym uruchamianiu systemu. Również pamięciowy system plików jest często montowany w katalogu `/tmp`. Proces ten może zostać zautomatyzowany wykorzystując zmienne `rc.conf(5)` związane z `tmpmfs` (bądź za pomocą wpisu w `/etc/fstab`; patrz `mdmfs(8)`).

Większość programów i aplikacji wykorzystywanych przez użytkowników.

Najczęściej używane programy, narzędzia programistyczne, aplikacje.

Pliki nagłówkowe C.

Biblioteki.

Pliki danych różnych programów użytkowych.

Demony i programy systemowe (uruchamiane przez inne programy).

Lokalne programy, biblioteki, itp. Ponadto jest to domyślny katalog dla instalowanych portów. Ogólna struktura katalogów wewnątrz `/usr/local` powinna odpowiadać strukturze `/usr` opisanej w dokumentacji hier(7). Wyjątkiem jest katalog `man`, umieszczony bezpośrednio w `/usr/local`, a nie w `/usr/local/share`, oraz dokumentacja portów, znajdująca się w `share/doc/port`.

Pliki zależne od architektury komputera, tworzone w procesie budowania drzewa `/usr/src`.

Kolekcja portów FreeBSD (opcjonalna).

Demony i programy systemowe (dostępne dla użytkowników).

Pliki niezależne od architektury systemu.

Katalog

```

/usr/src/
/usr/X11R6/

/var/

/var/log/
/var/mail/
/var/spool/
/var/tmp/

/var/yp

```

Opis

Pliki źródłowe BSD, lokalne pliki źródłowe.

Pliki wykonywalne, biblioteki, i inne pliki dystrybucji X11R6 (opcjonalnie).

Różne pliki dzienników systemowych, pliki tymczasowe, pliki kolejek. Również pamięciowy system plików jest często montowany w tym katalogu. Proces ten może zostać zautomatyzowany wykorzystując zmienne `rc.conf(5)` związane z `varmfs` (bądź za pomocą wpisu w `/etc/fstab`; patrz `mdmfs(8)`).

Pliki dzienników systemowych.

Skrzynki pocztowe użytkowników.

Katalogi kolejek systemu drukowania i poczty.

Pliki tymczasowe nie usuwane przy ponownym uruchamianiu systemu.

Mapy usługi NIS.

3.5. Organizacja dysku

Najmniejszą jednostką organizacji dysku używaną przez FreeBSD do odnajdywania plików jest nazwa pliku. W nazwach plików rozróżniane są duże i małe litery, tak więc `readme.txt` i `README.TXT` to dwa różne pliki. FreeBSD nie wykorzystuje rozszerzeń nazw plików (`.txt`) do określenia, czy plik jest programem, dokumentem, czy innym zbiorem danych.

Pliki przechowywane są w katalogach. Katalog może być pusty, lub może zawierać setki plików. Może również zawierać inne katalogi, dzięki czemu mamy możliwość zbudowania hierarchicznej struktury katalogów. Pozwala to na łatwą organizację danych.

Dostęp do plików i katalogów uzyskuje się podając nazwę pliku lub katalogu, poprzedzoną ukośnikiem `/` i innymi wymaganymi nazwami katalogów. Jeśli mamy katalog `foo`, a w nim katalog `bar`, w którym znajduje się plik `readme.txt`, wówczas pełną nazwą, bądź ścieżką dostępu do pliku jest `foo/bar/readme.txt`.

Katalogi i pliki przechowywane są w systemie plików. Każdy system plików ma jeden katalog najwyższego poziomu, zwany *katalogiem głównym* systemu plików. W katalogu głównym mogą być umieszczone następne katalogi.

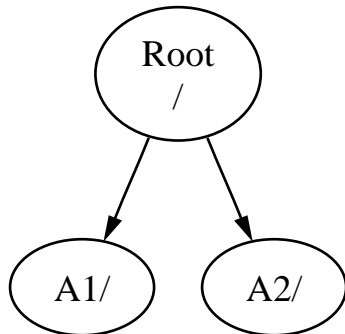
To, o czym mówimy, jest zapewne podobne do innych systemów operacyjnych, z którymi być może zetknęliśmy się wcześniej. Są jednak różnice; na przykład w systemie MS-DOS nazwy plików i katalogów oddzielane są znakiem `\`, w Mac OS® natomiast znakiem `:`.

We FreeBSD nie są używane litery dysków, lub inne nazwy dysków w ścieżce. Nie spotkamy się w FreeBSD z czymś takim jak `c:/foo/bar/readme.txt`.

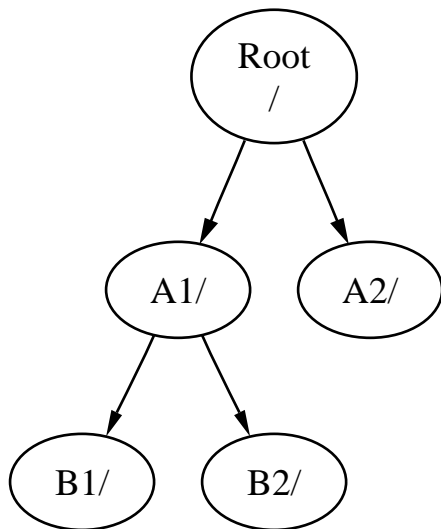
Jest natomiast jeden system plików pełniący rolę *głównego systemu plików*. Zawiera on katalog główny dostępny jako `/`. Każdy inny system plików jest *montowany* w głównym systemie plików. Niezależnie od tego, ile dysków mamy w komputerze, we FreeBSD każdy katalog wydaje się być częścią tego samego dysku.

Założmy, że mamy trzy systemy plików, nazwane A, B i C. Każdy z nich ma katalog główny, zawierający dwa katalogi o nazwach A1, A2 (oraz odpowiednio B1, B2 i C1, C2).

Niech A będzie głównym systemem plików. Gdybyśmy sprawdzili jego zawartość poleceniem `ls`, zobaczylibyśmy dwa podkatalogi A1 i A2. Drzewo katalogów wygląda następująco:

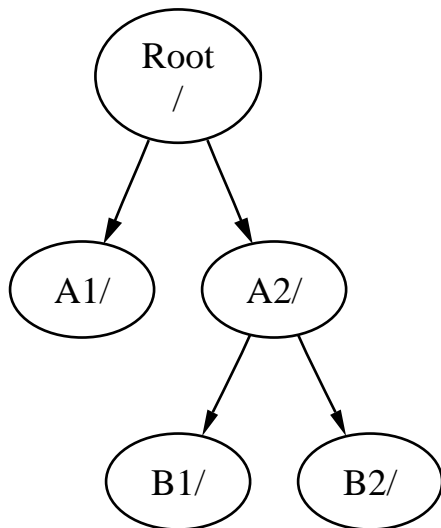


System plików musi być montowany w katalogu innego systemu plików. Przyjmijmy teraz, że montujemy system plików B w katalogu A1. Główny katalog B zastąpi A1, a podkatalogi B pojawią się w odpowiednim miejscu:



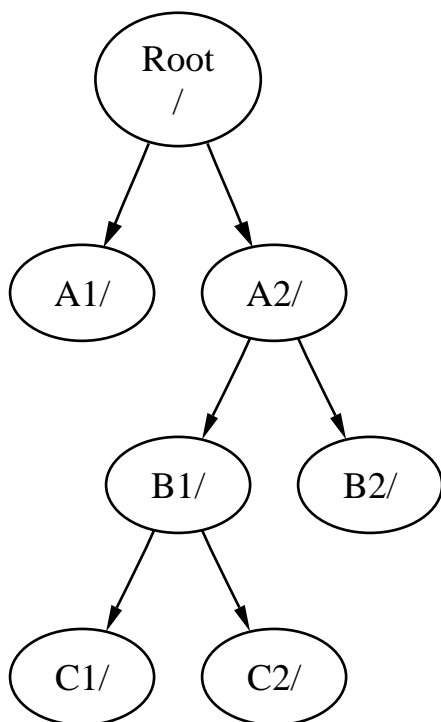
Do plików znajdujących się w katalogach B1 i B2 można się dostać posługując się ścieżką `/A1/B1` lub `/A1/B2`. Pliki poprzednio obecne w katalogu `/A1` są tymczasowo ukryte. Pojawią się ponownie po *odmontowaniu* B z A.

Gdyby zamontować B w A2, drzewo katalogów wyglądałoby tak:

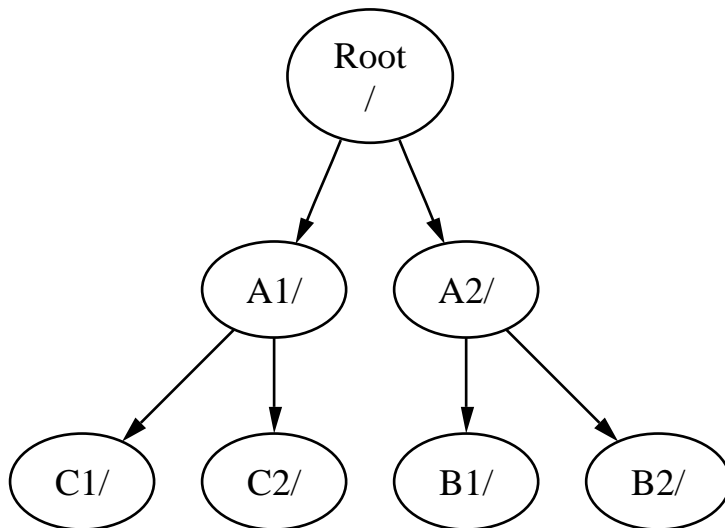


ścieżki natomiast miałyby postać `/A2/B1` i `/A2/B2`.

Systemy plików mogą być montowane jeden na drugim. Rozwijając poprzedni przykład, możemy zamontować system plików C w katalogu B1 systemu plików B, otrzymując następującą postać drzewa katalogów:



Można równie dobrze zamontować C bezpośrednio w systemie plików A, w katalogu A1:



Znającym system MS-DOS może to przypominać polecenie `join`, choć nie jest to to samo.

Zwykle nie trzeba zajmować się opisanymi powyżej rzeczami. Najczęściej tworzymy systemy plików podczas instalacji FreeBSD, wybieramy miejsce ich zamontowania i nie wprowadzamy później żadnych zmian, chyba, że zainstalujemy nowy dysk.

Można utworzyć jeden obszerny główny system plików i nie tworzyć żadnych innych. Takie podejście ma kilka wad i jedną zaletę.

Korzyści z kilku systemów plików

- Odrębne systemy plików mogą mieć różne *opcje montowania* (mount options). Na przykład, przy odpowiednim przygotowaniu, główny system plików może być zamontowany tylko do odczytu, przez co niemożliwe będzie przypadkowe usunięcie lub zmiana ważnego pliku. Oddzielenie systemów plików dostępnych do zapisu dla użytkowników, jak np. `/home`, od innych pozwala również na montowanie ich z opcją *nosuid*; co z kolei pozwala zwiększyć bezpieczeństwo systemu uniemożliwiając wykorzystanie bitów *suid/guid*.
- FreeBSD automatycznie optymalizuje układ plików w systemie plików, w zależności od tego, jak ów system jest wykorzystywany. System plików zawierający wiele często zapisywanych małych plików będzie optymalizowany inaczej niż taki, w którym przechowywane jest mniej plików o dużych rozmiarach. W przypadku jednego dużego systemu plików taka optymalizacja nie zadziała.
- Systemy plików FreeBSD są odporne na awarie zasilania. W niesprzyjających okolicznościach może się jednak zdarzyć, że przerwa w dostawie prądu w krytycznym momencie spowoduje uszkodzenie struktury systemu plików. Przechowywanie danych w kilku systemach plików zwiększa szansę, że system uruchomi się ponownie, dzięki czemu łatwiej będzie odzyskać dane z kopii zapasowej.

Korzyść z pojedynczego systemu plików

- Systemy plików mają stały rozmiar. Podczas instalacji FreeBSD tworzymy system plików o zadanym rozmiarze; później może się okazać, że trzeba powiększyć partycję. Niełatwo jest to zrobić inaczej, niż przez przygotowanie zapasowej kopii danych, utworzenie na nowo systemu plików o większych rozmiarach, oraz skopiowanie danych z powrotem.

WAŻNE: We FreeBSD dostępne jest polecenie `growfs(8)`, które pozwala na zwiększenie rozmiaru systemu plików w locie, pomijając wspomniane ograniczenie.

Systemy plików przechowywane są na partycjach. Pojęcie partycji ma tu inne znaczenie niż popularnie stosowane (np. partycja systemu MS-DOS), ze względu na uniksowy rodowód FreeBSD. Każda z partycji oznaczana jest literą, od a do h. Pojedyncza partycja może zawierać jeden system plików, dlatego też do systemów plików często odwołuje się albo poprzez miejsce ich zamontowania w głównym systemie plików, albo przez literowe oznaczenie partycji, na której dany system plików się znajduje.

Przestrzeń dyskowa jest również używana we FreeBSD jako *przestrzeń wymiany*, pełniąc w ten sposób rolę *pamięci wirtualnej*. Komputer może dzięki temu dysponować większą ilością pamięci, niż ma w rzeczywistości. Kiedy pamięci zaczyna brakować, FreeBSD odsyła niektóre nieużywane dane do przestrzeni wymiany, a gdy znów okażą się potrzebne, przenosi je z powrotem (odsyłając jednocześnie inne dane).

Z niektórymi partycjami związane są pewne konwencje dotyczące ich zastosowania./para>

Partycja	Konwencja
a	Zwykle zawiera główny system plików
b	Zwykle zawiera przestrzeń wymiany
c	Zwykle jest tego samego rozmiaru, co obejmujący ją segment. Dzięki temu programy działające na całym segmencie (na przykład wykrywające uszkodzone obszary dysku) mogą działać na partycji c. Zwykle nie tworzy się na tej partycji systemu plików.
d	Swego czasu partycja d miała specjalne znaczenie, obecnie już go nie ma. Do dziś jednak niektóre programy mogą dziwnie się zachowywać, jeśli każe im się pracować na partycji d, dlatego też sysinstall zwykle w ogóle jej nie tworzy.

Każda partycja zawierająca system plików przechowywana jest na czymś, co we FreeBSD nosi nazwę *segmentu*. Jest to określenie tego, co wcześniej zwane było partycją, i ponownie jest to konsekwencją uniksowych korzeni FreeBSD. Segmenty są oznaczane liczbami od 1 do 4.

Numery segmentów, wraz z przedrostkiem *s*, poprzedzone są nazwą urządzenia. Tak więc „da0s1” jest pierwszym segmentem na pierwszym dysku SCSI. Na dysku mogą być najwyżej cztery fizyczne segmenty, można jednak tworzyć segmenty logiczne wewnątrz segmentów fizycznych specjalnego typu. Powstałe w ten sposób segmenty rozszerzone mają numery od 5 wzwyż, zatem „ad0s5” odpowiada pierwszemu rozszerzonemu segmentowi na dysku IDE. Urządzenia te są wykorzystywane przez systemy plików, które zajmują cały segment.

Segmenty, dyski „niebezpiecznie dedykowane” i inne dyski zawierają *partycje*, oznaczane literami od a do h. Litera dopisywana jest do nazwy urządzenia, więc „da0a” odpowiadać będzie partycji a na pierwszym dysku da, „niebezpiecznie dedykowanym”. Z kolei „ad1s3e” oznacza piątą partycję w trzecim segmencie drugiego dysku IDE.

Własne oznaczenie ma także każdy dysk. Nazwa dysku składa się z symbolu określającego typ dysku, oraz numeru, określającego który to dysk. Dyski, inaczej niż segmenty, numerowane są od zera. Tabela 3-1 zawiera najczęściej spotykane zwykle oznaczenia.

Gdy odwołujemy się do partycji, FreeBSD wymaga, byśmy podali również nazwę obejmującego ją segmentu i dysku. Z kolei gdy odwołujemy się do segmentu, podajemy również nazwę dysku. Kolejno podajemy więc nazwę dysku, *s*, numer segmentu, a na koniec literę partycji; patrz Przykład 3-1.

Przykład 3-2 pokazuje schematyczny model dysku, z pomocą którego łatwiej będzie zrozumieć pewne rzeczy.

Gdy instalujemy FreeBSD, w pierwszej kolejności musimy przygotować segmenty na dysku, następnie w segmencie przeznaczonym dla FreeBSD utworzyć partycję, następnie wewnątrz partycji stworzyć system plików (lub przestrzeń wymiany) i określić miejsce jego montowania.

Tabela 3-1. Oznaczenia dysków

Oznaczenie	Znaczenie
ad	Dysk ATAPI (IDE)
da	Dysk SCSI o dostępie bezpośrednim
acd	CDROM ATAPI (IDE)
cd	CDROM SCSI
fd	Stacja dyskietek

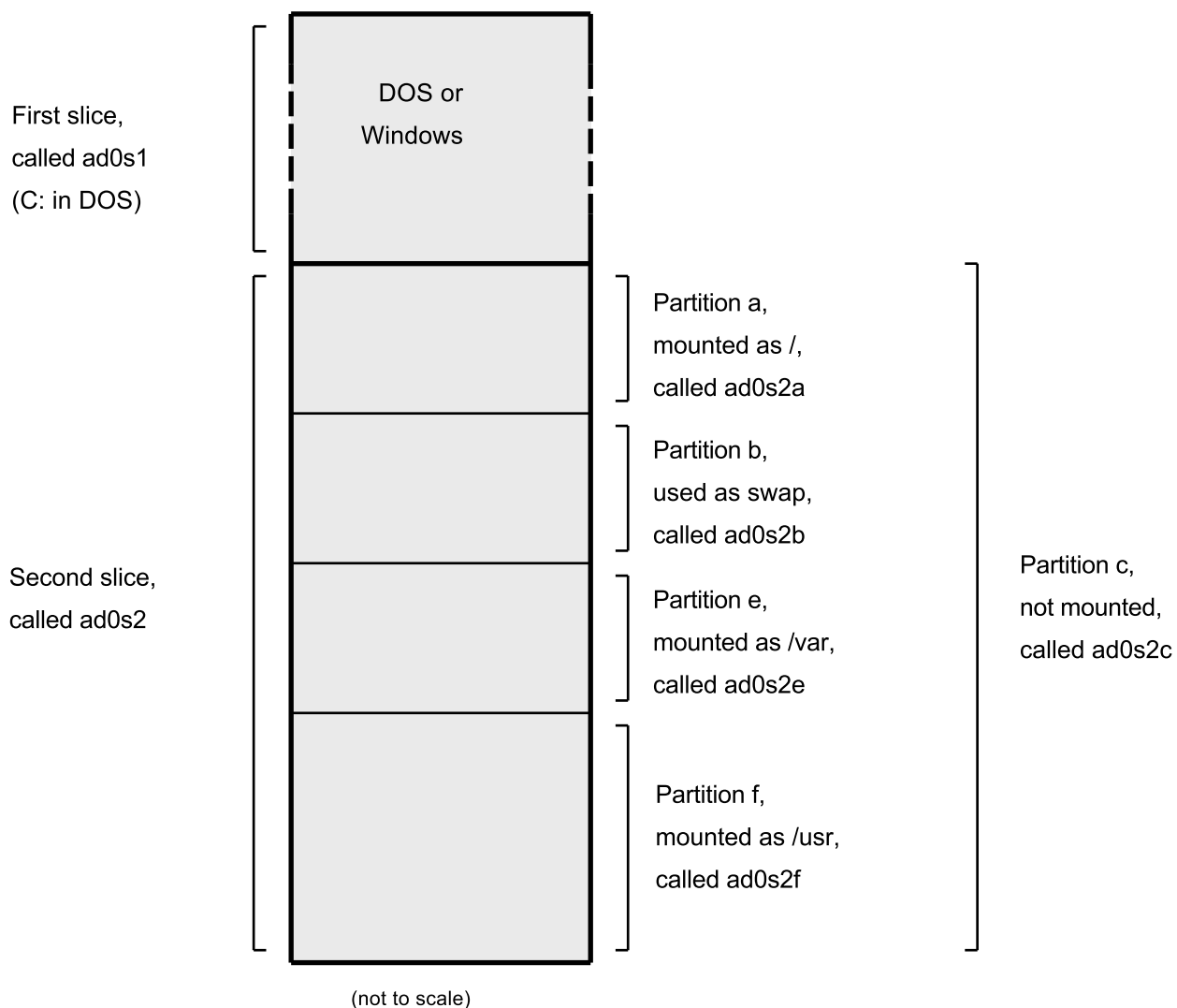
Przykład 3-1. Przykładowe nazwy dysków, segmentów i partycji

Nazwa	Znaczenie
ad0s1a	Pierwsza partycja (a) w pierwszym segmencie (s1) na pierwszym dysku IDE (ad0).
da1s2e	Piąta partycja e w drugim segmencie (s2) na drugim dysku SCSI (da1).

Przykład 3-2. Schematyczny model dysku

Rysunek przedstawia pierwszy dysk IDE z punktu widzenia FreeBSD. Zakładamy, że dysk ma rozmiar 4 GB i jest podzielony na dwa segmenty (partycje w MS-DOS) o rozmiarze po 2 GB. Pierwszy segment zawiera DOS-owy dysk C:, natomiast w drugim segmencie znajduje się przykładowa instalacja FreeBSD, z trzema partycjami oraz partycją wymiany.

Każda z trzech partycji przechowuje system plików. Na partycji a umieszczony jest główny system plików, na e znajduje się katalog /var, a na f katalog /usr.



3.6. Montowanie i odmontowywanie systemów plików

System plików można sobie wyobrazić jako drzewo, którego korzeniem jest /. /dev, /usr i inne podkatalogi katalogu głównego są gałęziami, z których mogą wyrastać kolejne gałęzie, na przykład /usr/local, itd.

Jest kilka powodów, dla których warto jest trzymać niektóre katalogi w oddzielnych systemach plików. W katalogu /var znajdują się podkatalogi log/ i spool/ oraz rozmaite pliki tymczasowe, z tego powodu może się on zapełnić. Zapełnienie głównego systemu plików jest raczej niepożądane, więc często zaleca się oddzielenie /var od /.

Często niektóre katalogi umieszczane są na odrębnych systemach plików ze względu na to, że znajdują się na osobnych dyskach fizycznych lub dyskach wirtualnych, jak na przykład pliki udostępniane poprzez Network File System lub napędy CDROM.

3.6.1. Plik `fstab`

Systemy plików wymienione w pliku `/etc/fstab` są automatycznie montowane podczas ładowania systemu (prócz oznaczonych opcją `noauto`).

Wpisy w pliku `/etc/fstab` są następującej postaci:

<i>urządzenie</i>	<i>/punkt-montowania</i>	<i>typ</i>	<i>opcje</i>	<i>archiwizacja</i>	<i>nr-przebiegu</i>
-------------------	--------------------------	------------	--------------	---------------------	---------------------

urządzenie

Nazwa pliku urządzenia (istniejącego), zgodnie z opisem w Sekcja 17.2.

punkt-montowania

Katalog (istniejący), w którym system plików ma być zamontowany.

typ

Typ systemu plików przekazywany poleceniu `mount(8)`. We FreeBSD domyślnie jest to `ufs`.

opcje

Pierwszą opcją jest `rw`, jeśli w systemie plików ma być możliwy odczyt i zapis, albo `ro`, jeżeli dozwolony ma być tylko odczyt. W następnej kolejności podawane są inne opcje. Często stosowana jest opcja `noauto`, która zapobiega automatycznemu montowaniu systemu plików podczas uruchamiania systemu. Pozostałe opcje opisane są w dokumentacji systemowej `mount(8)`.

archiwizacja

Na podstawie tej informacji program `dump(8)` stwierdza, które systemy plików mają być archiwizowane. Jeśli pole to zostanie pominięte, domyślnie przyjmowana jest wartość zero.

nr-przebiegu

Na podstawie tego pola wyznaczana jest kolejność, w jakiej systemy plików poddawane są sprawdzaniu. Systemy plików, które nie mają być sprawdzane, powinny mieć `nr-przebiegu` ustawiony na zero. Główny system plików (powinien być sprawdzony jako pierwszy) powinien mieć `nr-przebiegu` o wartości jeden, a inne systemy plików powinny mieć wpisaną wartość większą od jednego. Jeśli dwa lub więcej systemów plików będzie miało taki sam `nr-przebiegu`, wówczas `fsck(8)`, o ile będzie to możliwe, podejmie próbę równoległego sprawdzenia tych systemów plików.

Więcej informacji o formacie pliku `/etc/fstab` oraz definiowanych w nim opcji dostępnych w podręczniku systemowym `fstab(5)`

3.6.2. Polecenie `mount`

Polecenie `mount(8)` jest głównym poleceniem używanym do montowania systemów plików.

W najprostszej postaci, używa się go następująco:

mount urządzenie punkt-montowania

Polecenie to ma mnóstwo opcji wymienionych w dokumentacji systemowej mount(8). Do najczęściej stosowanych należą:

Opcje montowania

-a

Montowanie wszystkich systemów plików wymienionych w `/etc/fstab`. Nie są montowane systemy plików z opcją „noauto” oraz wykluczone przez opcję `-t`, jak również systemy plików już zamontowane.

-d

Wykonanie wszystkiego, oprócz faktycznego wywołania funkcji systemowej montowania. W połączeniu z opcją `-v` można w ten sposób sprawdzić, co tak naprawdę mount(8) stara się zrobić.

-f

Wymuszenie montowania nieuporządkowanego systemu plików (niebezpieczne), lub wymuszenie odebrania prawa do zapisu przy zmianie trybu montowania systemu plików z trybu „odczyt i zapis” na „tylko do odczytu”.

-r

Montowanie systemu plików w trybie tylko do odczytu. Taki sam efekt ma zastosowanie opcji `-o` z argumentem `ro` (bądź `rdonly` w wersjach FreeBSD wcześniejszych niż 5.2).

-t typ

Montowanie systemu plików o określonym typie. Przy zastosowaniu opcji `-a` montowane są tylko systemy plików podanego typu.

Domyślnym typem systemu plików jest „ufs”.

-u

Uaktualnienie opcji montowania systemu plików.

-v

Pokazywanie dodatkowych komunikatów.

-w

Montowanie w trybie odczytu i zapisu.

Opcji `-o` towarzyszy lista oddzielonych przecinkami parametrów, oto niektóre z nich:

nodev

Ignorowanie obecnych w systemie plików urządzeń specjalnych. Przydatna opcja, jeśli chodzi o bezpieczeństwo.

noexec

Wyłączenie uruchamiania programów wykonywalnych na systemie plików. Również służy bezpieczeństwu.

nosuid

Ignorowanie bitów setuid i setgid w systemie plików. Kolejna opcja służąca bezpieczeństwu.

3.6.3. Polecenie `umount` Command

Poleceniu `umount(8)` należy podać jako parametr punkt montowania, nazwę urządzenia bądź opcję `-a` lub `-A`.

Każdej z form wywołania polecenia można podać opcję `-f`, która nakazuje dokonać bezwarunkowego odmontowania, oraz opcję `-v`, powodującą wypisywanie dodatkowych komunikatów. Należy mieć na uwadze, że raczej nie zaleca się korzystania z `-f`. Bezwarunkowe odmontowywanie systemu plików może doprowadzić do awarii systemu lub uszkodzenia danych znajdujących się w danym systemie plików.

Opcje `-a` oraz `-A` służą do odmontowania wszystkich zamontowanych systemów plików, lub systemów plików wybranych typów, określonych w opcji `-t`. Opcja `-A` nie dokonuje próby odmontowania głównego systemu plików.

3.7. Procesy

FreeBSD jest wielozadaniowym systemem operacyjnym. Oznacza to, że korzystając z systemu mamy wrażenie, że wiele programów działa jednocześnie. Działający w danej chwili program nazywany jest *procesem*. Po wydaniu dowolnego polecenia uruchamiany jest przynajmniej jeden proces. Są również procesy systemowe, które działają nieprzerwanie, zapewniając prawidłowe funkcjonowanie systemu.

Każdemu procesowi przypisany jest jednoznaczny numer zwany *identyfikatorem procesu*, lub po prostu *PID*. Podobnie jak plik, również każdy proces ma swojego właściciela i grupę. Na podstawie informacji o właścicielu i grupie system operacyjny przydziela procesowi prawa do otwierania plików i urządzeń, przy zastosowaniu opisanych wcześniej praw dostępu. Większość procesów ma swój proces macierzysty; jest to proces, który uruchomił dany proces. Przykładowo, kiedy wydajemy polecenia w powłoce, to zarówno powłoka jest procesem, jak i każde z wykonanych poleceń. Procesem macierzystym każdego uruchomionego w ten sposób procesu będzie powłoka. Wyjątkiem jest specjalny proces zwany `init(8)`. `init` jest pierwszym procesem, więc jego PID jest zawsze równy 1. Proces `init` uruchamiany jest przez jądro systemu podczas ładowania FreeBSD.

Są dwa bardzo przydatne polecenia, które pozwalają zobaczyć, jakie procesy są uruchomione: `ps(1)` i `top(1)`. Polecenie `ps` pokazuje statyczną listę działających w danej chwili procesów, uwzględniając informacje takie jak PID-y procesów, zużywaną pamięć, wydane do uruchomienia procesów polecenia, itd. Polecenie `top` wyświetla listę uruchomionych procesów, która jest co kilka sekund uaktualniana, dzięki czemu możemy na bieżąco śledzić, czym zajmuje się komputer.

Domyślnie `ps` pokazuje tylko działające procesy należące do użytkownika wydającego polecenie. Na przykład:

```
% ps
  PID  TT  STAT      TIME COMMAND
   298  p0  Ss      0:01.10 tcsh
  7078  p0  S        2:40.88 xemacs mdoc.xsl (xemacs-21.1.14)
37393  p0  I        0:03.11 xemacs freebsd.dsl (xemacs-21.1.14)
48630  p0  S        2:50.89 /usr/local/lib/netscape-linux/navigator-linux-4.77.bi
48730  p0  IW       0:00.00 (dns helper) (navigator-linux-)
72210  p0  R+       0:00.00 ps
   390  p1  Is       0:01.14 tcsh
  7059  p2  Is+      1:36.18 /usr/local/bin/mutt -y
```

```

6688  p3  IWs    0:00.00 tcsh
10735  p4  IWs    0:00.00 tcsh
20256  p5  IWs    0:00.00 tcsh
  262  v0  IWs    0:00.00 -tcsh (tcsh)
  270  v0  IW+    0:00.00 /bin/sh /usr/X11R6/bin/startx -- -bpp 16
  280  v0  IW+    0:00.00 xinit /home/nik/.xinitrc -- -bpp 16
  284  v0  IW     0:00.00 /bin/sh /home/nik/.xinitrc
  285  v0  S      0:38.45 /usr/X11R6/bin/sawfish

```

Jak widzimy, `ps(1)` wyświetla informacje w kilku kolumnach. W kolumnie `PID` pokazywany jest omówiony wcześniej identyfikator procesu. `PID`-y są przydzielane po kolei od 1 do 99999 i znów od początku, gdy się skończą. Kolumna `TT` pokazuje terminal, na którym działa program — na razie nie będziemy się tym zajmować. W kolumnie `STAT` przedstawiony jest stan procesu, jego także na razie nie będziemy omawiać. `TIME` pokazuje czas wykorzystywania procesora przez dany proces, niekoniecznie odpowiada on czasowi, jaki upłynął od uruchomienia programu, ponieważ wiele programów przez długi czas oczekuje na jakieś zdarzenie, a dopiero potem wykorzystuje procesor. Ostatnia kolumna, `COMMAND`, pokazuje polecenie, którym uruchomiony został program.

`ps(1)` ma wiele rozmaitych opcji, które mają wpływ na wyświetlane informacje. Jedną z najbardziej przydatnych kombinacji opcji jest `auxww`. Opcja `a` pokazuje informacje o wszystkich działających procesach, również nie należących do nas. `u` pokazuje nazwę użytkownika, do którego należy proces, jak również wykorzystanie pamięci. `x` pokazuje informacje o procesach — demonach. Opcja `ww` nakazuje, by polecenie `ps(1)` wyświetlało pełną linię polecenia, nie obcinając jej, by zmieściła się na ekranie.

Informacje pokazywane przez `top(1)` wyglądają podobnie. Oto przykład:

```

% top
last pid: 72257;  load averages:  0.13,   0.09,   0.03    up 0+13:38:33  22:39:10
47 processes:  1 running, 46 sleeping
CPU states: 12.6% user,   0.0% nice,   7.8% system,   0.0% interrupt, 79.7% idle
Mem: 36M Active, 5256K Inact, 13M Wired, 6312K Cache, 15M Buf, 408K Free
Swap: 256M Total, 38M Used, 217M Free, 15% Inuse

  PID USERNAME PRI NICE  SIZE  RES STATE   TIME  WCPU   CPU COMMAND
72257 nik       28   0 1960K 1044K RUN     0:00 14.86% 1.42% top
 7078 nik        2   0 15280K 10960K select  2:54  0.88%  0.88% xemacs-21.1.14
  281 nik        2   0 18636K  7112K select  5:36  0.73%  0.73% XF86_SVGA
  296 nik        2   0  3240K 1644K select  0:12  0.05%  0.05% xterm
48630 nik        2   0 29816K  9148K select  3:18  0.00%  0.00% navigator-linu
  175 root        2   0   924K  252K select  1:41  0.00%  0.00% syslogd
 7059 nik        2   0  7260K 4644K poll   1:38  0.00%  0.00% mutt
...

```

Informacje podzielone są na dwie części. Nagłówek (pierwsze pięć wierszy) zawiera `PID` ostatnio uruchomionego procesu, średnie obciążenie systemu (miara zapracowania systemu), czas działania systemu (od ostatniego uruchomienia) oraz aktualny czas. Inne liczby w nagłówku informują o liczbie działających procesów (w przykładzie 47), jak dużo pamięci i przestrzeni wymiany jest zajęte, oraz ile czasu system przebywa w różnych stanach procesora.

Pod nagłówkiem w kilku kolumnach pokazane są informacje zbliżone do przedstawianych przez `ps(1)`. Podobnie można tu znaleźć `PID` procesu, nazwę użytkownika, czas zajmowania procesora, oraz polecenie, którym uruchomiono proces. `top(1)` pokazuje domyślnie także rozmiar pamięci zajmowanej przez proces. Ta ostatnia informacja podzielona jest na dwie kolumny; jedna odpowiada całkowitemu rozmiarowi, druga rozmiarowi rezydentnemu. Całkowity rozmiar oznacza, ile pamięci było potrzebne programowi, z kolei rozmiar rezydentny

informuje, ile pamięci wykorzystuje program w danej chwili. W przykładzie widać, że **Netscape®** potrzebował prawie 30 MB pamięci RAM, jednak obecnie wykorzystuje tylko 9 MB.

top(1) automatycznie aktualizuje wyświetlane informacje co dwie sekundy; można to zmienić opcją `s`.

3.8. Demony, sygnały i unicestwianie procesów

Kiedy korzystamy z edytora tekstu, możemy go w prosty sposób obsługiwać, wczytywać pliki, itp. Jest to możliwe dzięki cechom samego edytora oraz dzięki temu, że edytor jest podłączony do *terminala*. Jednakże, niektóre programy pracują bez ciągłej komunikacji z użytkownikiem, są więc odłączone od terminala. Przykładem takiego programu może być serwer WWW, nieustannie odpowiadający na żądania pochodzące z sieci, bez potrzeby komunikacji z użytkownikiem. Inny przykład to programy przesyłające emaile pomiędzy komputerami.

Takie programy nazywane są *demonami* (ang. daemons). Demony to postaci z mitologii greckiej — niewielkie usługowe istoty, ani dobre, ani złe, które w rozmaity sposób pomagały ludziom. Podobnie pomagają dzisiejsze serwery pocztowe i serwery WWW. Dlatego właśnie od długiego czasu maskotką BSD jest wesoły demon z widłami i w

Przyjęto, iż programy uruchamiane jako demony mają nazwy zakończone literą „d”. **BIND** (Berkeley Internet Name Daemon) jest serwerem nazw uruchamianym przez program `named`, serwer WWW **Apache** nosi nazwę `httpd`, demon kolejowania drukarki (line printer spooling daemon) to `lpd`, itd. Nie jest to sztywna reguła, lecz przyjęta konwencja; na przykład główny demon pocztowy programu **Sendmail** nazywa się `sendmail`, a nie jak można by przypuszczać `maild`.

Niekiedy istnieje potrzeba komunikacji z procesem — demonem. Odbywa się ona poprzez *sygnały*, to znaczy możemy porozumieć się z demonem (lub jakimkolwiek działającym procesem) wysyłając mu sygnał. Są różne rodzaje sygnałów, które możemy wysłać — niektóre z nich mają określone znaczenie, inne są odpowiednio interpretowane przez aplikację, co powinno być opisane w dokumentacji aplikacji. Sygnał możemy wysłać tylko do procesu, którego jesteśmy właścicielem. Wysłanie sygnału do procesu należącego do kogoś innego za pośrednictwem `kill(1)` lub `kill(2)` spowoduje odmowę dostępu. Wyjątkiem jest użytkownik `root`, który może wysyłać sygnały do dowolnego procesu, niezależnie od jego właściciela.

Zdarza się, że samo FreeBSD również wysyła aplikacjom sygnały. Jeżeli niewłaściwie napisany program próbuje dostać się do niedostępnego dla niego obszaru pamięci, FreeBSD wysyła procesowi sygnał *Segmentation Violation* (`SIGSEGV`). Aplikacja może skorzystać z funkcji systemowej `alarm(3)`, wówczas po upływie pewnego czasu zostanie do niej wysłany sygnał Alarm (`SIGALRM`). I tak dalej.

Do zatrzymania procesu można wykorzystać dwa sygnały: `SIGTERM` i `SIGKILL`. Pierwszy z nich jest łagodnym sposobem unicestwienia procesu; proces może *przechwycić* ten sygnał, następnie zakończyć swoją pracę, np. zamykając pliki, które otworzył. Czasami proces może zignorować sygnał `SIGTERM`, jeśli akurat zajmuje się czymś, co nie powinno być przerywane.

Sygnał `SIGKILL` nie może zostać zignorowany. Działa według zasady „Nie obchodzi mnie, co robisz, w tej chwili przestań”. Wysłanie procesowi sygnału `SIGKILL` powoduje, iż FreeBSD natychmiast go wstrzymuje⁴.

Inne użyteczne sygnały to `SIGHUP`, `SIGUSR1` i `SIGUSR2`. Są to sygnały ogólnego przeznaczenia, różne aplikacje reagują na nie w różny sposób.

Powiedzmy, że dokonaliśmy zmiany w pliku konfiguracji serwera WWW, i chcemy nakazać serwerowi, aby konfiguracja została ponownie wczytana. Moglibyśmy zatrzymać i ponownie uruchomić `httpd`, ale ubocznym efektem takiego postępowania byłaby chwilowa przerwa w pracy serwera, co jest raczej niepożądane. Większość demonów działa w taki sposób, iż po otrzymaniu sygnału `SIGHUP` dokonują ponownego przeczytania swojego pliku konfiguracyjnego. Dzięki temu zamiast unicestwiania i ponownego uruchamiania `httpd` możemy wysłać mu sygnał

SIGHUP. Nie jest jednoznacznie określone, jak procesy reagują na sygnał SIGHUP, dlatego różne demony mogą zachowywać się w różny sposób — w razie niepewności warto zapoznać się z dokumentacją konkretnego demona.

Sygnały wysyłane są przy użyciu polecenia `kill(1)`, jak w poniższym przykładzie.

Wysyłanie sygnału do procesu

W tym przykładzie zaprezentowano wysyłanie sygnału do `inetd(8)`. Plik konfiguracyjny dla `inetd` to `/etc/inetd.conf`. Wysłanie sygnału SIGHUP spowoduje ponowne przeczytanie tego pliku.

1. Trzeba ustalić PID procesu, do którego wysyłać będziemy sygnał — do tego celu posłużą polecenia `ps(1)` i `grep(1)`. Polecenia `grep(1)` używamy do odnalezienia podanego ciągu znaków. Ponieważ polecenia wydajemy jako zwykły użytkownik, a `inetd(8)` działa jako `root`, polecenie `ps(1)` musimy wywołać z opcją `ax`.

```
% ps -ax | grep inetd
 198  ??   IwS      0:00.00 inetd -wW
```

Jak widać, `inetd(8)` ma PID o wartości 198. Niekiedy w przedstawionym powyżej przykładzie może się także pojawić proces `grep inetd`, wynika to ze sposobu, w jaki `ps(1)` odnajduje działające procesy.

2. Sygnał wysyłamy przy pomocy polecenia `kill(1)`. Najpierw skorzystamy jednak z polecenia `su(1)` by stać się rootem, gdyż `inetd(8)` działa jako `root`.

```
% su
Password:
# /bin/kill -s HUP 198
```

Podobnie jak wiele poleceń w systemach UNIX, `kill(1)` nie wyświetla żadnego komunikatu w przypadku powodzenia. Jeżeli natomiast sygnał został wysłany do procesu, którego nie jest się właścicielem, pojawi się informacja: `kill: PID: Operation not permitted` (niedozwolona operacja). Błędne wpisanie PID-u spowoduje albo wysłanie sygnału do niewłaściwego procesu, co może skończyć się źle, albo też wysłanie sygnału do PID-u, który nie jest w danej chwili wykorzystywany — pojawi się wówczas komunikat `kill: PID: No such process` (nie ma takiego procesu).

Dlaczego warto korzystać z `/bin/kill`? W wielu powłokach polecenie `kill` jest wbudowane; oznacza to, że sama powłoka zajmuje się wysyłaniem sygnału, nie wywołując `/bin/kill`. Może to być użyteczne, jednakże w różnych powłokach stosowana jest różna składnia do określenia nazwy sygnału, który ma być wysłany. Zamiast więc zapamiętywania wszystkich możliwych składni, łatwiej jest po prostu korzystać z polecenia `/bin/kill ...`.

Inne sygnały wysyła się tą samą metodą, wystarczy zastąpić `TERM` lub `KILL` w odpowiedni sposób.

WAŻNE: Unicestwienie losowo wybranego procesu jest raczej złym pomysłem. Szczególne znaczenie ma `init(8)`, proces o PID równym 1. Wydanie polecenia `/bin/kill -s KILL 1` jest szybką metodą wyłączenia systemu. Należy zawsze sprawdzać poprawność argumentów polecenia `kill(1)` przed naciśnięciem klawisza **Return**.

3.9. Powłoki

W codziennej pracy z FreeBSD bardzo często wykorzystywany jest interfejs linii poleceń, zwany powłoką (ang. shell). Podstawowym zadaniem powłoki jest przyjmowanie poleceń i wykonywanie ich. Wiele powłok wyposażonych jest także w dodatkowe funkcje ułatwiające pracę, np. usprawnienia zarządzania plikami, dopasowywanie nazw plików, ułatwienia korzystania z linii poleceń, makropolecenia i zmienne środowiskowe. We FreeBSD dostępnych jest kilka powłok, np. Bourne Shell `sh` i ulepszony C-shell `tcsh`. Wiele innych powłok, jak choćby `zsh` czy `bash`, można znaleźć w kolekcji portów FreeBSD.

Którą z powłok najlepiej jest używać? To właściwie kwestia gustu. Dla programistów C najwygodniejsze mogą być powłoki o składni wzorowanej na języku C, np. `tcsh`. Użytkownikom Linuksa i tym, dla których interfejs linii poleceń systemów 8unix; jest nowością, można polecić `bash`. Do wyboru jest wiele powłok, każda z nich ma pewne charakterystyczne tylko dla niej właściwości, które niekoniecznie będą działać w każdym warunkach.

Często spotykanym udogodnieniem powłoki jest uzupełnianie nazw plików. Po wpisaniu kilku pierwszych liter polecenia lub nazwy pliku powłoka potrafi zwykle uzupełnić dalszy ciąg polecenia lub nazwy, dzieje się to po wciśnięciu klawisza **Tab**. Przyjmijmy przykładowo, że istnieją pliki o nazwach `foobar` i `foo.bar`. Chcemy usunąć plik `foo.bar`. Możemy więc wydać polecenie: `rm fo[Tab].[Tab]`.

Powłoka wyświetli: `rm foo[BEEP].bar`.

Napis [BEEP] oznacza sygnał dźwiękowy, będący informacją od powłoki, że uzupełnienie nazwy pliku nie było możliwe, ponieważ można dopasować więcej niż jedną nazwę. Zarówno `foobar` jak i `foo.bar` zaczynają się od `fo`. Powłoka mogła jednakże uzupełnić początek, czyli `foo`. Teraz można wpisać kropkę `.` i ponownie wcisnąć **Tab**, tym razem powłoka uzupełni nazwę do końca.

Inną cechą powłoki są zmienne środowiskowe. Przechowywane są one w przestrzeni środowiska powłoki w postaci par „nazwa = wartość”. Przestrzeń środowiska jest widoczna dla każdego programu uruchamianego przez powłokę, dlatego też przechowuje się tam wiele parametrów konfiguracyjnych dla programów. Oto najczęściej spotykane zmienne środowiskowe wraz z krótkim opisem:

Zmienna	Opis
USER	Nazwa aktualnie zalogowanego użytkownika.
PATH	Lista katalogów zawierających pliki wykonywalne oddzielona przecinkami.
DISPLAY	Nazwa ekranu X11, jeśli takowy jest dostępny.
SHELL	Wykorzystywana powłoka.
TERM	Nazwa terminala użytkownika, wykorzystywana do określenia właściwości terminala.
TERMCAP	Zapis z bazy termcap zawierający sekwencje kodów odpowiadających różnym funkcjom terminala.
OSTYPE	Typ systemu operacyjnego, np. FreeBSD.
MACHTYPE	Architektura sprzętowa, na jakiej działa system.
EDITOR	Preferowany przez użytkownika edytor tekstu.
PAGER	Preferowany przez użytkownika program wyświetlający pliki tekstowe.
MANPATH	Lista katalogów zawierających dokumentację systemową oddzielona przecinkami.

Sposób odczytywania i ustawiania zmiennych środowiskowych zależy od rodzaju używanej powłoki. Na przykład w powłokach wzorowanych na C, jak `tcsh` i `csh`, do ustawiania i przeglądania zmiennych środowiskowych służy polecenie `setenv`, natomiast w powłokach Bourne'a, czyli `sh` i `bash`, do tych celów wykorzystywane jest polecenie `export`. Przykładowo, aby zmienić zmienną środowiskową `EDITOR` na `/usr/local/bin/emacs` w powłoce `csh` lub `tcsh`, należy wydać polecenie:

```
% setenv EDITOR /usr/local/bin/emacs
```

A w powłokach Bourne'a:

```
% export EDITOR="/usr/local/bin/emacs"
```

W większości powłok można wyświetlić wartość zmiennej środowiskowej przez poprzedzenie jej nazwy znakiem `$`. Dla przykładu, polecenie `echo $TERM` pokaże wartość zmiennej `$TERM`, ponieważ powłoka zastępuje wyrażenie `$TERM` wartością zmiennej i przekazuje ją do `echo`.

Wiele znaków, zwanych meta-znakami, traktowanych jest przez powłoki w specjalny sposób. Najczęściej wykorzystywanym jest `*`, oznaczający dowolny ciąg znaków w nazwie pliku, umożliwiający wykonywanie operacji na wielu plikach. Przykładowo, wywołanie `echo *` jest prawie identyczne z wywołaniem `ls`, ponieważ powłoka przekazuje do `echo` nazwy wszystkich plików pasujących `*`.

Jeśli potrzeba, by powłoka nie interpretowała znaku jako znak specjalny, należy go poprzedzić znakiem ukośnika (`\`). Wywołanie `echo $TERM` powoduje wypisanie ustawionego typu terminala, podczas gdy efektem polecenia `echo \$TERM` jest po prostu napis `$TERM`.

3.9.1. Zmiana powłoki

Najłatwiej jest zmienić powłokę przy użyciu polecenia `chsh`. Wywołanie tego polecenia uruchomi edytor wskazany przez zmienną `EDITOR`, lub edytor `vi`, jeśli nie jest ona zdefiniowana. Następnie należy zmienić nazwę powłoki w wierszu „Shell:”.

Można też skorzystać z `chsh` z opcją `-s`, która automatycznie zmieni powłokę, bez uruchamiania edytora. Poniżej przedstawiono wywołanie zmieniające powłokę na `bash`:

```
% chsh -s /usr/local/bin/bash
```

Notatka: Wybrana powłoka *musi* być wymieniona w pliku `/etc/shells`. Jeśli powłokę zainstalowano z kolekcji portów powinna zostać dopisana automatycznie. Jeśli natomiast przeprowadzono ręczną instalację powłoki, trzeba to zrobić samemu.

Dla przykładu, jeśli powłoka `bash` została zainstalowana i umieszczona w `/usr/local/bin`, trzeba będzie wydać polecenie:

```
# echo "/usr/local/bin/bash" >> /etc/shells
```

Oraz uruchomić `chsh`.

3.10. Edytory tekstu

Tłumaczył Aleksander Fafuła.

Konfiguracja FreeBSD polega głównie na edytowaniu plików tekstowych. Z tego właśnie powodu, dobrze byłoby zapoznać się z edytorami tekstu. FreeBSD posiada ich kilka, a kolejne można doinstalować z drzewa portów.

Najłatwiejszym do nauki i w użyciu jest edytor **ee**, co jest skrótem od Easy Editor (ang. Łatwy Edytor). Aby uruchomić **ee**, należy użyć polecenia `ee plik`, gdzie *plik* jest to, co chcemy edytować. Na przykład, aby wyedytować plik `/etc/rc.conf`, napiszemy `ee /etc/rc.conf`. Gdy już jesteśmy w **ee**, możemy zauważyć, że wszystkie niezbędne komendy są wypisane u góry ekranu. Znak `^` oznacza wciśnięty klawisz **Ctrl**. Innymi słowy `^e` oznacza, że należy trzymać **Ctrl** i wcisnąć klawisz **e**. Aby wyjść z **ee**, wciśnij **Esc**, następnie wybierz `leave editor` (opuść edytor). Edytor zapyta, czy zachować zmiany, jeśli plik został zmodyfikowany.

FreeBSD w swoich zasobach ma także potężny edytor tekstu, jakim jest **vi**. W kolekcji portów dostępny jest także **Emacs**, czy **vim** (`editors/emacs` i `editors/vim`). Edytory te oferują dużo większą funkcjonalność, ale oczekują w zamian większego opanowania użytkownika z zasadami ich działania, ponadto ich obsługa jest trudniejsza do nauki. Jednakże, jeśli planujesz edytować wiele tekstu, nauka **Emacs** lub **vim** zwróci się w długim okresie w postaci zaoszczędzonego czasu.

3.11. Urządzenia i pliki urządzeń

Mianem urządzeń określa się komponenty komputera, takie jak dysk, drukarka, karta graficzna czy klawiatura. Podczas ładowania systemu FreeBSD większość wyświetlanych komunikatów dotyczy wykrywanych urządzeń. Komunikaty startowe dostępne są do późniejszego przeglądania w pliku `/var/run/dmesg.boot`.

Przykładowo, `acd0` odpowiada pierwszemu napędowi CDROM IDE, natomiast `kbd0` oznacza klawiaturę.

Dostęp do większości urządzeń w systemie operacyjnym UNIX odbywa się poprzez specjalne pliki, zwane plikami urządzeń, znajdujące się w katalogu `/dev`.

3.11.1. Tworzenie plików urządzeń

Kiedy wyposażamy komputer w nowe urządzenie, lub kompilujemy jądro z obsługą dodatkowych urządzeń, konieczne może okazać się utworzenie nowych plików urządzeń.

3.11.1.1. DEVFS (DEVice File System)

System plików urządzeń, zwany DEVFS, udostępnia przestrzeń nazw urządzeń jądra jako część przestrzeni nazw głównego systemu plików. DEVFS zajmuje się obsługą systemu plików urządzeń, dzięki czemu nie trzeba samodzielnie tworzyć bądź modyfikować plików urządzeń.

Więcej informacji znaleźć można w dokumentacji systemowej `devfs(5)`.

3.12. Formaty binarne

Tłumaczył Cezary Morga.

By zrozumieć czemu FreeBSD używa formatu elf(5), musimy wpierw poznać trzy obecnie „dominujące” formaty plików wykonywalnych w systemach UNIX:

- a.out(5)

Najstarszy i najbardziej „klasyczny” format w Uniksie. Wykorzystuje krótki nagłówek z magicznym numerem na samym początku, często wykorzystywanym do określenia rodzaju pliku (szczegółowy opis dostępny jest w a.out(5)). Na plik składają się trzy segmenty: .text, .data i .bss oraz tablice symboli i ciągów tekstowych.

- COFF

Format obiektowy pochodzący z SVR3. W tym formacie sekcja tablic wchodzi już w skład nagłówka, tak więc możliwe jest zawarcie w pliku więcej sekcji niż tylko .text, .data i .bss.

- elf(5)

Następca COFF zawierający wiele dodatkowych sekcji o 32- bądź nawet 64-bitowych wartościach. Jednym, acz wielkim minusem jest fakt, iż przy projektowaniu formatu ELF również założono, że na każdą architekturę sprzętową będzie istniał tylko jeden interfejs ABI. Okazało się natomiast, iż takie założenie jest błędne nawet w świecie komercyjnych SYSV (z którego pochodzą przynajmniej trzy ABI: SVR4, Solaris i SCO).

Sposobem na rozwiązanie tego problemu we FreeBSD są narzędzia do *metkowania* plików wykonywalnych ELF informacjami, z którymi ABI jest on zgodny. Więcej informacji dostępnych jest w podręczniku systemowym brandelf(1).

System FreeBSD pochodzi z „klasycznego” obozu. Wykorzystywał on zatem format a.out(5) — technologię wypróbowaną w wielu pokoleniach systemów BSD i z powodzeniem stosowaną aż do gałęzi 3.X. Mimo, że skompilowanie i uruchomienie w sposób natywny plików binarnych ELF (a także jądra) było możliwe we FreeBSD już od pewnego czasu, Projekt oficjalnie opierał się przed migracją do formatu ELF jako podstawowego. Dlaczego? Otóż, gdy obóz linuksowy wykonał ten bolesny krok ku ELF nie udało się tak łatwo uciec od formatu a.out. Wynikało to przede wszystkim z faktu, iż niezbyt elastyczny plan migracji bazował na mechanizmie współdzielonych bibliotek, których modyfikacja narażała wielu trudności zarówno producentom sprzętu jak i projektantom. Dopiero od momentu gdy narzędzia dostępne dla ELF zaoferowały sposób rozwiązania problemu ze współdzielonymi bibliotekami, zaczęły być postrzegane ogólnie jako „droga do przodu”, a tym samym koszty migracji mogły zostać uznane za niezbędne do poniesienia. Mechanizm współdzielonych bibliotek FreeBSD w dużej mierze przypomina mechanizm z SunOS™ Sun’a i jako taki jest bardzo łatwy w użyciu.

Skąd więc tyle różnych formatów?

W zamierzonych czasach do dyspozycji był prosty sprzęt komputerowy. Ów prosty sprzęt obsługiwał mały, prosty system. Stąd też format a.out był całkowicie odpowiednim do prezentacji plików binarnych w tym prostym systemie (PDP-11). Gdy UNIX został przeniesiony z tego prostego systemu na platformy typu Motorola 68k czy VAXen, zachowany został format a.out, zdecydowanie wystarczający dla wczesnych wersji Uniksa.

Pewien czas później, jakiś bystry inżynier sprzętowy stwierdził że gdyby potrafił zmusić oprogramowanie do robienia kilku obskurnych sztuczek, wówczas mógłby pozbyć się kilku bramek z układu scalonego i zmusić CPU do szybszej pracy. Pomimo, że format a.out potrafił współpracować z tym nowym rodzajem sprzętu (zwanego wówczas RISC) to mimo wszystko nie był najlepszym do tego formatem. Dlatego też rozpoczęto prace nad innymi formatami binarnymi, które miały osiągnąć lepsze wyniki niż ograniczony, prosty a.out mógł zaoferować. Stworzone zostały COFF, ECOFF oraz kilka mniej znanych formatów, nim powstał ELF.

Kolejnym problemem okazał się wzrost rozmiarów programów przy względnie małej pojemności dysków oraz pamięci fizycznych, a także zwiększeniu stopnia skomplikowania pamięci wirtualnej VM. Tak też narodziła się koncepcja współdzielonych bibliotek. Mimo, że ów postęp osiągnięty był przy pomocy formatu `a.out` zakres jego przydatności był stale rozciągany, wraz z każdą nową funkcją. Pojawiła się konieczność dynamicznego wczytywania pewnych rzeczy już w trakcie uruchamiania programu czy zapisywania części programu zaraz po wykonaniu kodu `init` w pamięci lub przestrzeni wymiany. Również języki programowania stawały się coraz bardziej wyrafinowane. Wiele poprawek wprowadzonych do formatu `a.out` umożliwiały realizację kolejnych funkcji, przy czym z reguły działały one tylko przez pewien czas. Niestety, format `a.out` stał się z czasem niezdolny do rozwiązywania wszystkich problemów bez wciąż rozrastającego się narzutu w kodzie i poziomu skomplikowania. Mimo, że ELF potrafił rozwiązać wiele z ówczesnych problemów, zmiana formatu binarnego, który generalnie działał, wciąż była wielką uciążliwością. Dlatego też ELF musiał poczekać aż bardziej bolesnym okazało się pozostanie przy `a.out` niż przejście do ELF.

Wraz z upływem czasu, narzędzia kompilacyjne, z których FreeBSD wywodzi własne narzędzia (przede wszystkim assembler i loader), wyewoluowały w dwa równoległe projekty. Odmiana FreeBSD dała współdzielone biblioteki oraz poprawki kilku błędów. Ludzie z GNU, którzy oryginalnie napisali te programy, przepisali je na nowo i dodali proste kompilatory wskrośne, pozwalające na pracę w różnych formatach. Nowy pakiet narzędzi GNU (**binutils**) wspiera kompilowanie wskrośne, format ELF, współdzielone biblioteki, rozszerzenia C++, itp. Dodatkowo, wielu producentów sprzętu przygotowuje binaria ELF. Jest to zatem dobra rzecz dla FreeBSD, że je obsługuje.

Format ELF oferuje większą rozszerzalność niż `a.out`. Narzędzia ELF są lepiej przygotowywane i oferują kompilację wskrośną, co jest istotne dla wielu programistów. Co prawda ELF może być trochę wolniejszy niż `a.out`, jednakże próba pomiaru może być trudna. Istnieje również wiele innych szczegółów różnych dla obydwu formatów, m.in. sposób mapowania stron, obsługi kodu `init` itp. Co prawda, żadne z nich nie jest istotne, jednakże różnice istnieją. Z czasem, wsparcie dla `a.out` zostanie wstrzymane z jądra **GENERIC** i ostatecznie usunięte z jądra gdy tylko zniknie potrzeba obsługi programów `a.out`.

3.13. Więcej informacji

3.13.1. Dokumentacja systemowa

Najdokładniejszą dokumentacją we FreeBSD jest dokumentacja systemowa. Dla prawie każdego dostępnego w systemie programu przygotowana jest krótka instrukcja obsługi, omawiająca podstawy jego działania i rozmaite opcje. Dokumentację możemy przeglądać przy pomocy polecenia `man`. Korzystanie z tego polecenia jest bardzo proste:

```
% man polecenie
```

`polecenie` jest nazwą polecenia, o którym chcemy uzyskać informacje. Na przykład, aby dowiedzieć się czegoś na temat polecenia `ls` wpisujemy:

```
% man ls
```

Dokumentacja systemowa podzielona jest na ponumerowane części:

1. Polecenia dostępne dla użytkowników.
2. Funkcje systemowe i kody błędów.
3. Funkcje z bibliotek języka C.

4. Sterowniki urządzeń.
5. Formaty plików.
6. Gry i inne rozrywki.
7. Różne informacje.
8. Polecenia służące do zarządzania systemem.
9. Informacje dla programistów jądra.

Niekiedy takie samo zagadnienie może pojawić się w kilku częściach dokumentacji. Na przykład istnieje polecenie `chmod`, oraz funkcja systemowa `chmod()`. W taki wypadku możemy wybrać interesującą nas część dokumentacji, podając jej numer jako parametr polecenia `man`:

```
% man 1 chmod
```

W efekcie pokazana zostanie dokumentacja polecenia `chmod`. Zgodnie z przyjętą konwencją, numer odpowiedniej części dokumentacji podawany jest w nawiasach, tak więc `chmod(1)` odpowiada poleceniu `chmod`, natomiast `chmod(2)` odpowiada funkcji systemowej.

W opisany powyżej sposób możemy dowiedzieć się, jak korzystać z danego polecenia, jeśli znamy jego nazwę. Co zrobić, jeśli nie możemy sobie przypomnieć nazwy polecenia? Otóż, `man` potrafi również wyszukiwać wybranych słów kluczowych w opisach poleceń, służy do tego opcja `-k`:

```
% man -k mail
```

Wpisanie takiego polecenia spowoduje wyświetlenie listy poleceń, których opisy zawierają słowo kluczowe „mail”. Takie działanie jest równoważne skorzystaniu z polecenia `apropos`.

Jeśli więc, przeglądając zawartość katalogu `/usr/bin`, zastanawiamy się, do czego właściwie służą znajdujące się tam polecenia, możemy wpisać:

```
% cd /usr/bin
% man -f *
```

lub

```
% cd /usr/bin
% whatis *
```

W obu przypadkach efekt będzie taki sam.

3.13.2. Pliki GNU Info

Do FreeBSD dołączonych jest wiele programów i narzędzi stworzonych przez Free Software Foundation (FSF). Prócz dokumentacji systemowej, do tych programów dołączone są bardziej rozbudowane dokumenty hipertekstowe, zwane plikami `info`. Można je przeglądać poleceniem `info`, lub trybem `info` **emacsa**, o ile **emacs** został zainstalowany.

By skorzystać z polecenia `info(1)`, wpisujemy:

```
% info
```

Krótkie wprowadzenie pojawia się po wpisaniu `h`. Spis poleceń jest dostępny po wpisaniu `?`.

Przypisy

1. Takie jest znaczenie symbolu `i386`. Zwróćmy uwagę, że nawet wówczas, gdy FreeBSD działa na procesorze Intela innym niż 386, w tym miejscu znajdzie się napis `i386`. Nie określa on bowiem typu używanego procesora, lecz jego „architekturę”.
2. Skrypty startowe to programy uruchamiane automatycznie podczas ładowania FreeBSD. Ich podstawowym zadaniem jest przygotowanie środowiska pracy dla innych programów, oraz uruchomienie wybranych usług działających w tle, pełniących różne przydatne funkcje.
3. Szczegółowy opis obecnych we FreeBSD sterowników konsoli i klawiatury można znaleźć w dokumentacji systemowej `syscons(4)`, `atkbd(4)`, `vidcontrol(1)` i `kbdcontrol(1)`. Nie będziemy tutaj zajmować się szczegółami, zainteresowani czytelnicy są jak najbardziej zachęceni do zapoznania się z dokumentacją systemową, w której omawiane teraz zagadnienia opisane są dokładniej.
4. Nie do końca jest to prawdą — w kilku przypadkach nie można przerwać procesu. Na przykład gdy proces stara się przeczytać plik znajdujący się na innym komputerze w sieci, a ów inny komputer z jakiegoś powodu będzie niedostępny (na skutek awarii sieci, lub po prostu zostanie wyłączony), to proces stanie się „nieprzerwalny”. Po chwili (zwykle po dwóch minutach) proces przekroczy czas oczekiwania, wówczas zostanie unicestwiony.

Rozdział 4. Instalacja programów: pakiety i porty

Tłumaczył Cezary Morga.

4.1. Streszczenie

System FreeBSD rozprowadzany jest wraz z bogatą kolekcją narzędzi systemowych. Tym nie mniej, stanowi to absolutne minimum. Szybko pojawia się bowiem potrzeba zainstalowania dodatkowego oprogramowania, by móc rozpocząć prawdziwą pracę z systemem. FreeBSD dostarcza dwóch dopełniających się metod instalacji oprogramowania: kolekcję portów FreeBSD (kompilacja programów ze źródeł) i system pakietów (instalacja z gotowych binariów). Każda z tych metod może zostać wykorzystana do instalacji najnowszych wersji ulubionego oprogramowania z lokalnych nośników bądź bezpośrednio z sieci.

Przeczytawszy ten rozdział dowiemy się:

- Jak instalować oprogramowanie innych producentów dostarczane w postaci binarnej.
- Jak kompilować oprogramowanie innych producentów z wykorzystaniem kolekcji portów.
- Jak usunąć poprzednio zainstalowane pakiety bądź porty.
- Jak zmienić domyślne wartości wykorzystywane przy kompilacji portów.
- Jak odnaleźć właściwe oprogramowanie.
- Jak zaktualizować wykorzystywane aplikacje.

4.2. Omówienie instalacji oprogramowania

Osoby, które już wcześniej pracowały z systemami UNIX wiedzą, że typowy proces instalacji oprogramowania sprowadza się mniej więcej do następujących punktów:

1. Pobranie programu, który może być rozprowadzany w postaci kodu źródłowego bądź binarnej.
2. Rozpakowania programu z formatu w jakim jest rozprowadzany (najczęściej jest to plik tar skompresowany za pomocą `compress(1)`, `gzip(1)` lub `bzip2(1)`).
3. Odnalezienie dokumentacji (najczęściej plik `INSTALL` lub `README` bądź pliki w podkatalogu `doc/`) i zapoznanie się z instrukcjami instalacji programu.
4. Kompilacja programu, jeśli rozprowadzany jest w postaci źródłowej. Może to wymagać również wykonania dodatkowych czynności, jak np. edycji pliku `Makefile` bądź uruchomienia skryptu `configure`.
5. Weryfikacja i instalacja aplikacji.

Wszystko to przy założeniu, że w między czasie nie pojawiły się żadne trudności. Instalacja oprogramowania, które nie było przygotowywane z myślą o FreeBSD może wymagać nawet modyfikacji kodu źródłowego nim zacznie poprawnie funkcjonować.

Oczywiście, we FreeBSD można instalować oprogramowanie „tradycyjnym” sposobem. Jednakże system ten posiada dwa rozwiązania, które potrafią zaoszczędzić mnóstwo czasu i trudu: pakiety i porty. W chwili pisania tego tekstu, dostępnych za pomocą tych systemów jest przeszło 23,000 aplikacji.

Dla każdego programu dostępny jest do pobrania pojedynczy pakiet, który zawiera skompilowane kopie plików aplikacji, zarówno plików uruchomieniowych jak i konfiguracyjnych czy dokumentacji. Pobranym plikiem można manipulować za pomocą poleceń `pkg_add(1)`, `pkg_delete(1)`, `pkg_info(1)`, itp. Nowe programy można instalować za pomocą zaledwie jednego polecenia.

Port natomiast, jest zbiorem plików mających za zadanie zautomatyzować proces kompilacji danego programu z kodu źródłowego.

O ile typowa kompilacja programu składa się z wielu czynności wykonywanych przez użytkownika, o tyle pliki składające się na port zawierają dostateczną ilość informacji aby pozwolić systemowi zrobić to za nas. Wystarczy wprowadzić kilka prostych poleceń a system automatycznie pobierze kod źródłowy programu, rozpakuje, nałoży łatki, skompiluje i zainstaluje za nas.

Ponadto system portów może również posłużyć do przygotowania pakietów, którymi następnie można manipulować za pomocą `pkg_add` i innymi poleceniami zarządzających pakietami.

Obydwa systemy potrafią analizować *zależności* występujące pomiędzy aplikacjami. Załóżmy, że chcemy zainstalować program, który zależy od pewnej biblioteki. Zarówno program jak i biblioteka dostępne są w systemach portów i pakietów FreeBSD. Niezależnie od tego czy wykorzystamy polecenie `pkg_add` czy porty, by zainstalować program, to obydwie systemy spostrzegą, że biblioteka nie została zainstalowana i automatycznie zainstalują najpierw bibliotekę.

Można by się zastanawiać dlaczego FreeBSD wykorzystuje obydwie systemy, skoro ich działanie jest tak bardzo podobne. Tak pakiety jak i porty posiadają pewne zalety. Który system wykorzystamy zależy od naszych własnych upodobań.

Zalety pakietów

- Skompresowany plik pakietu zajmuje z reguły mniej miejsca niż skompresowany plik zawierający kod źródłowy.
- Instalacja pakietów nie wymaga dodatkowej kompilacji. W przypadku dużych aplikacji, jak np. **Mozilla**, **KDE** czy **GNOME** może to być istotne. Szczególnie gdy pracuje się na dość wolnej maszynie.
- Stosowanie pakietów nie wymaga żadnej wiedzy o procesie kompilowania oprogramowania w systemie FreeBSD.

Zalety portów

- Pakiety są z reguły kompilowane z dość typowymi opcjami, ponieważ powinny być przydatne do wykorzystania na maksymalnej liczbie komputerów. Instalując programy z portów mamy możliwość „podkręcenia” opcji kompilacji, by (przykładowo) skompilować program zoptymalizowany dla procesorów Pentium IV lub Athlon.
- Niektóre aplikacje posiadają pewne opcje kompilacji związane z zadaniami, które mają realizować. Przykładowo **Apache** może zostać skompilowany z wieloma różnorodnymi opcjami. Kompilując go z portów nie musimy zgadzać się na domyślne opcje mogąc samemu dokonać wyboru.

W niektórych przypadkach dostępnych jest kilka pakietów tej samej aplikacji skompilowanych z różnymi parametrami. Na przykład program **Ghostscript** dostępny jest jako pakiet `ghostscript` oraz `ghostscript-nox11`, zależnie od tego czy mamy zainstalowany serwer X11. O ile tego typu rozwiązania są teoretycznie możliwe do zrealizowania w systemie pakietów, o tyle staje się to praktycznie niemożliwe gdy aplikacja posiada więcej niż kilka różnych opcji kompilacji.

- Warunki licencji niektórych aplikacji zabraniają rozprowadzania w postaci binarnej. Muszą być zatem rozprowadzane jako kod źródłowy.
- Niektórzy nie ufają pakietom binarnym. W przypadku kodu źródłowego można (przynajmniej w teorii) przejrzeć go i samemu poszukać potencjalnych luk.
- Jeśli posiadamy własne łaty będziemy potrzebowali kodu źródłowego aby je nanieść do programu.
- Jeszcze inni po prostu lubią mieć pod ręką kod źródłowy, by móc go poczytać gdy się nudzą, zmodyfikować czy zapożyczyć pewne rozwiązania (o ile pozwala na to licencja), itd.

Najlepszym sposobem śledzenia zmian dokonywanych w systemie portów jest zapisanie się na Listę dyskusyjną portów FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports>) oraz Listę dyskusyjną błędów w systemie portów FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs>).

Ostrzeżenie Przed instalacją jakiegokolwiek aplikacji należy sprawdzić na stronie <http://vuxml.freebsd.org/> czy w danym programie istnieją luki związane bezpieczeństwem.

Alternatywnie możemy zainstalować `security/portaudit`, który automatycznie sprawdza wszystkie instalowane programy pod względem znanych luk bezpieczeństwa; weryfikowane są również porty przed kompilacją. W międzyczasie można wykorzystać polecenie `portaudit -F -a`, by sprawdzić zainstalowane już pakiety.

Pozostała część niniejszego rozdziału ma za zadanie wyjaśnić jak z wykorzystaniem systemu pakietów i portów instalować w systemie FreeBSD oprogramowanie innych producentów.

4.3. Odnalezienie programu dla siebie

Nim przystąpimy do instalacji programów musimy wiedzieć co chcemy zainstalować i jak się nazywa.

Lista dostępnych we FreeBSD programów rośnie cały czas. Na szczęście jest wiele sposobów na odnalezienie tego czego szukamy:

- Na stronie internetowej FreeBSD pod adresem <http://www.FreeBSD.org/ports/> (<http://www.FreeBSD.org/ports/index.html>) znajduje jest aktualna lista dostępnych programów. Listę można dowolnie przeszukiwać według kilku kryteriów, np. nazwy (jeśli ją znamy). Możliwe jest również przejrzanie spisu wszystkich aplikacji znajdujących się w danej kategorii.
- Dzięki stronie FreshPorts (<http://www.FreshPorts.org/>) prowadzonej przez Dana Langille'a możliwe jest bieżące śledzenie zmian aplikacji w drzewie portów. Witryna umożliwia otrzymywanie informacji drogą emailową o zmianach w wybranych portach.
- Jeśli nie znamy nazwy programu, który chcemy zainstalować, warto poszukać go na stronach pokroju FreshMeat (<http://www.freshmeat.net/>) a następnie sprawdzić na stronie FreeBSD czy został przygotowany odpowiedni port.
- Jeśli znamy dokładną nazwę portu a chcemy sprawdzić z jakiej pochodzi kategorii, można skorzystać z polecenia `whereis(1)`. Wystarczy wpisać `whereis plik`, gdzie `plik` jest nazwą programu, którego poszukujemy. Otrzymany wynik będzie postaci:

```
# whereis lsof
lsof: /usr/ports/sysutils/lsof
```

Przykład ten informuje nas, że program `lsof` (narzędzie systemowe) znajduje się w katalogu `/usr/ports/sysutils/lsof`.

- Jeszcze innym sposobem na odnalezienie danego portu jest wykorzystanie mechanizmu przeszukiwania kolekcji portów. By skorzystać z tej funkcji należy przejść do katalogu `/usr/ports`. Następnie wpisać `make search name=nazwa-programu`, gdzie `program-name` jest nazwą poszukiwanej aplikacji. Przykładowo, szukając `lsof`:

```
# cd /usr/ports
# make search name=lsof
Port:    lsof-4.56.4
Path:    /usr/ports/sysutils/lsof
Info:    Lists information about open files (similar to fstat(1))
Maint:   obrien@FreeBSD.org
Index:   sysutils
B-deps:
R-deps:
```

Część wyniku, która nas interesuje to wiersz zaczynający się od „Path:”, a określający lokalizację portu. Pozostałe z uzyskanych w ten sposób informacji nie zostaną tutaj opisane, gdyż nie są potrzebne do instalacji programu.

Szersze przeszukiwanie kolekcji portów możliwe jest wykorzystując `make search key=zwrot`, gdzie `zwrot` jest dowolnym wyrazem. Opcja ta przeszukuje nazwy portów, komentarze, opisy i listy zależności. Może być wykorzystana do odnalezienia portów związanych z danym zagadnieniem gdy nie znamy nazwy poszukiwanego programu.

W obydwu przypadkach nie są rozróżniane małe i duże litery w poszukiwanym ciągu. Szukając zatem „LSOF” oraz „lsof” otrzymamy takie same wyniki.

4.4. Korzystanie z systemu pakietów

Napisał Chern Lee. Tłumaczył Aleksander Fafuła. Przekład uzupełnił Cezary Morga.

4.4.1. Instalacja pakietów

Programu `pkg_add(1)` można użyć do instalacji programów zarówno z dysku lokalnego, jak i z sieci.

Przykład 4-1. Ręczne pobranie pakietu i instalacja lokalna

```
# ftp -a ftp2.FreeBSD.org
Connected to ftp2.FreeBSD.org.
220 ftp2.FreeBSD.org FTP server (Version 6.00LS) ready.
331 Guest login ok, send your email address as password.
230-
230-    This machine is in Vienna, VA, USA, hosted by Verio.
230-    Questions? E-mail freebsd@vienna.verio.net.
230-
230-
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub/FreeBSD/ports/packages/sysutils/
250 CWD command successful.
```

```
ftp> get lsof-4.56.4.tgz
local: lsof-4.56.4.tgz remote: lsof-4.56.4.tgz
200 PORT command successful.
150 Opening BINARY mode data connection for 'lsof-4.56.4.tgz' (92375 bytes).
100% |*****| 92375 00:00 ETA
226 Transfer complete.
92375 bytes received in 5.60 seconds (16.11 KB/s)
ftp> exit
# pkg_add lsof-4.56.4.tgz
```

Jeśli nie posiadamy lokalnego źródła programów (np na płytach CD FreeBSD), będzie Ci prawdopodobnie łatwiej użyć komendy `pkg_add(1)` z opcją `-r`. Spowoduje to, że program samodzielnie określi odpowiednią wersję oprogramowania dla naszej wersji systemu. Następnie pobierze odpowiedni plik z sieci oraz go zainstaluje.

```
# pkg_add -r lsof
```

W powyższym przykładzie program pobierze właściwy pakiet i zainstaluje go bez jakiejkolwiek dalszej ingerencji użytkownika. Jeśli chcemy wskazać programowi alternatywny serwer lustrzany, należy odpowiednio zdefiniować zmienną środowiskową `PACKAGESITE`. Program `pkg_add(1)` do pobierania plików z serwerów wykorzystuje `fetch(3)`, który z kolei wykorzystuje różnorodne zmienne środowiskowe, m.in. `FTP_PASSIVE_MODE`, `FTP_PROXY` oraz `FTP_PASSWORD`. Może się okazać, że będziemy musieli zdefiniować niektóre z nich jeśli nasz komputer znajduje się za zaporą ogniową, bądź musi korzystać z serwera pośredniczącego FTP/HTTP proxy. Więcej informacji znaleźć można w podręczniku systemowym programu `fetch(3)`. Warto zauważyć, iż w powyższym przykładzie jako nazwę pakietu podano jedynie `lsof` zamiast `lsof-4.56.4`. Przy zdalnym pobieraniu pakietów nie należy podawać numeru wersji pakietu. Program `pkg_add(1)` automatycznie pobierze najnowszą wersję aplikacji.

Notatka: Program `pkg_add(1)` pobierze najnowszą wersję aplikacji jedynie, gdy wykorzystujemy FreeBSD-CURRENT albo FreeBSD-STABLE. W przypadku -RELEASE pobrana zostanie wersja pakietu zbudowana dla danego wydania. Ograniczenie to można obejść modyfikując zmienną środowiskową `PACKAGESITE`. Na przykład, jeśli korzystamy z FreeBSD 5.4-RELEASE domyślnie `pkg_add(1)` będzie pobierał pakiety z `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-5.4-release/Latest/`. By zmusić go do pobierania pakietów zbudowanych dla FreeBSD 5-STABLE należy zmodyfikować zmienną `PACKAGESITE` by wskazywała na `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-5-stable/Latest/`.

Pakiety rozpowszechniane są w formacie `.tgz` oraz `.tbz`. Możemy je pobrać z

`ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/`, w Polsce z

`ftp://ftp.pl.FreeBSD.org/pub/FreeBSD/ports/packages/`, bądź odnaleźć na płytach CDROM FreeBSD. Każda płyta z cztero płytowej dystrybucji (także PowerPak'a itp) zawiera pakiety w katalogu `/packages`. Struktura katalogu podobna jest do drzewa portów `/usr/ports`. Każda kategoria ma swój własny katalog, ponadto każdy pakiet może zostać odnaleziony w katalogu `All` (Wszystkie).

Struktura katalogów pakietów jest identyczna względem struktury katalogów portów. Porty i pakiety kooperują za sobą, tworząc wspólnie cały system pakietów/portów.

4.4.2. Zarządzanie pakietami

Narzędziem służącym do przedstawienia informacji o zainstalowanych pakietach oraz wyświetlającym ich krótki opis jest `pkg_info(1)`.

```
# pkg_info
cvsup-16.1      A general network file distribution system optimized for CV
docbook-1.2     Meta-port for the different versions of the DocBook DTD
...
```

Program `pkg_version(1)` jest natomiast narzędziem, które podsumowuje wersje wszystkich zainstalowanych pakietów. Porównuje je następnie z tymi które znajdują się w drzewie portów.

```
# pkg_version
cvsup           =
docbook         =
...
```

Symbol w drugiej kolumnie określa wiek zainstalowanej wersji oprogramowania względem wersji odnalezionej w portach.

Symbol	Znaczenie
=	Wersja odnaleziona w portach jest identyczna./entry>
<	Wersja jest starsza, niż ta odnaleziona w portach.
>	Zainstalowana wersja jest nowsza, niż znaleziona w portach. (Prawdopodobnie lokalne drzewo portów nie zostało zaktualizowane.)
?	Zainstalowany pakiet nie może zostać odnaleziony w drzewie portów. (Może to mieć miejsce np. w sytuacji gdy zainstalowany port został usunięty z kolekcji portów, bądź zmienił nazwę.)
*	Istnieje wiele wersji tego programu.

4.4.3. Usuwanie pakietów

Aby usunąć uprzednio zainstalowane oprogramowanie użyj `pkg_delete(1)`.

```
# pkg_delete xchat-1.7.1
```

4.4.4. Dodatkowe informacje

Wszystkie informacje o pakietach znajdują się w katalogu `/var/db/pkg`. Lista zainstalowanych plików, a także opis każdej paczki można odnaleźć właśnie w tym katalogu.

4.5. Korzystanie z kolekcji portów

Źródło: Aleksander Fafuła. Przekład uzupełnił Cezary Morga.

Poniższy podrozdział dostarcza podstawowych informacji z zakresu używania kolekcji portów, w stopniu umożliwiającym instalowanie lub odinstalowywanie programów z własnego systemu. Szczegółowy opis parametrów

polecenia `make` i zmiennych środowiskowych dostępny jest w podręczniku systemowym `ports(7)`.

4.5.1. Pozyskanie kolekcji portów

Zanim zainstalujemy jakikolwiek port, musimy pobrać kolekcję portów, która w zasadzie jest zestawem plików `Makefiles`, łań i opisowych. Kolekcja znajduje się w katalogu `/usr/ports`.

W trakcie instalacji FreeBSD, **sysinstall** zapytał czy chcemy zainstalować kolekcję portów. Jeśli wybraliśmy nie, poniższe instrukcje pomogą nam własnoręcznie zainstalować kolekcję portów:

Metoda CVSup

Jest to prosta i szybka metoda pobrania kolekcji portów wykorzystująca system **CVSup**. Więcej informacji o **CVSup** dostępnych jest w podrozdziale Korzystanie z CVSup.

Bardzo ważnym jest, aby upewnić się, że katalog `/usr/ports` jest pusty nim po raz pierwszy uruchomimy **CVSup**! Jeśli posiadamy już kolekcję portów pozyskaną z innego źródła **CVSup** nie usunie nieużywanych plików łań.

1. Zainstaluj pakiet `net/cvsup-without-gui`:

```
# pkg_add -r cvsup-without-gui
```

Więcej informacji w podrozdziale Instalacja CVSup (Sekcja A.5.2).

2. Uruchom `cvsup`:

```
# cvsup -L 2 -h cvsup.FreeBSD.org /usr/share/examples/cvsup/ports-supfile
```

Warto zastąpić `cvsup.FreeBSD.org` adresem serwera CVSup zlokalizowanego bliżej nas. Kompletna lista serwerów lustrzanych dostępna jest w podrozdziale Serwery lustrzane CVSup (Sekcja A.5.7).

Notatka: Można wykorzystać własny plik `ports-supfile`, by np. uniknąć konieczności podawania adresu serwera **CVSup** z linii poleceń.

1. W takim wypadku, jako użytkownik `root`, skopiuj plik `/usr/share/examples/cvsup/ports-supfile` do innego katalogu, np. `/root` bądź własnego katalogu domowego.
2. Zmodyfikuj plik `ports-supfile`.
3. Zmień wpis `CHANGE_THIS.FreeBSD.org` na adres wybranego serwera lustrzanego **CVSup**. Kompletna lista serwerów lustrzanych dostępna jest w podrozdziale Serwery lustrzane CVSup (Sekcja A.5.7).
4. Teraz uruchom `cvsup` używając polecenia::

```
# cvsup -L 2 /root/ports-supfile
```

3. Późniejsze wpisanie polecenia `cvsup(1)` spowoduje sprawdzenie zmian dokonanych w kolekcji portów i aktualizację lokalnej wersji. Nie spowoduje to natomiast automatycznie ponownego skompilowania wykorzystywanych przez nas portów.

Metoda Portsnap

Portsnap jest alternatywnym systemem dystrybucji kolekcji portów. Po raz pierwszy został dołączony do FreeBSD 6.0. W starszych wersjach może zostać zainstalowany z pakietu `sysutils/portsnap`:

```
# pkg_add -r portsnap
```

Szczegółowe informacje o możliwościach programu dostępne są w podrozdziale Korzystanie z Portsnap.

1. Ten punkt możemy pominąć jeśli posiadamy FreeBSD 6.1-RELEASE bądź najnowszą wersję programu **Portsnap**. Przy pierwszym uruchomieniu programu portsnap(8) zostanie automatycznie utworzony katalog `/usr/ports`. W starszych wersjach programu wymagane jest własnoręczne utworzenie katalogu:

```
# mkdir /usr/ports
```

2. Pobierz skompresowaną migawkę kolekcji portów do katalogu `/var/db/portsnap`. Można następnie zakończyć połączenie z Internetem, jeśli jest taka potrzeba.

```
# portsnap fetch
```

3. Jeśli uruchamiany **Portsnap** po raz pierwszy należy rozpakować migawkę do katalogu `/usr/ports`:

```
# portsnap extract
```

Jeśli posiadamy już kolekcję portów w `/usr/ports` i jedynie ją aktualizujemy, wpisujemy polecenie:

```
# portsnap update
```

Metoda sysinstall

Metoda ta instaluje kolekcję portów z lokalnego nośnika posługując się programem **sysinstall**. Zainstalowana zostanie kopia kolekcji z dnia, w którym przygotowana została dana wersja FreeBSD. Jeśli dysponujemy połączeniem z Internetem powinniśmy zawsze stosować jedną z metod opisanych powyżej.

1. Uruchom **sysinstall** jako użytkownik **root** (`/stand/sysinstall` w wersjach FreeBSD starszych niż 5.2):

```
# sysinstall
```

2. Przejdź w dół, wybierz **Configure**, i naciśnij **Enter**.
3. Przejdź w dół, wybierz **Distributions** i naciśnij **Enter**.
4. Przejdź w dół do opcji **ports** i naciśnij **Spację**.
5. Przejdź do góry do opcji **Exit** i naciśnij **Enter**.
6. Ustaw wybrany przez siebie typ medium instalacji, jak np. płytę CDROM, serwer FTP, itd.
7. Przejdź do góry do opcji **Exit** i naciśnij **Enter**.
8. Naciśnij **X** by wyjść z programu **sysinstall**.

4.5.2. Instalacja Portów

Pierwsza rzecz o jakiej należy wspomnieć omawiając kolekcję portów, jest „szkielet”. Mówiąc w skrócie, szkielet portu jest minimalnym zestawem plików, które informują FreeBSD, jak poprawnie skompilować i zainstalować program. Każdy szkielet portu zawiera:

- Plik `Makefile`. Plik ten zawiera różne dane określające jak skompilować aplikację oraz gdzie ją zainstalować w systemie.

- Plik `distinfo` Plik ten zawiera informacje dotyczące plików, które muszą zostać pobrane, by skompilować port. Ponadto zawiera sumy kontrolne, na podstawie których `md5(1)` potrafi sprawdzić, czy pliki nie uległy uszkodzeniu w trakcie pobierania z sieci.
- Katalog `files`, który zawiera łatki pozwalające skompilować i zainstalować program w naszym systemie FreeBSD. Łatki są małymi plikami, w których określone są zmiany dotyczące konkretnych plików. Są to pliki tekstowe i po prostu mówią „Usuń linię 10” lub „Zmień linię 26 na to: ...”. Łatki są także znane jako „diffs” (ang. skrót od różnice) ponieważ są generowane przez program `diff(1)`.
Ten katalog może zawierać także inne pliki używane do kompilacji portu.
- Plik opisu `pkg-descr`. Jest to bardziej szczegółowy, nierzadko wieloliniowy opis programu.
- Plik listy `pkg-plist`. Jest to lista wszystkich plików, które zostaną zainstalowane przez port. Jest to także lista plików, które należy usunąć w przypadku odinstalowywania.

Niekiedy porty zawierają również inne pliki, jak na przykład `pkg-message` (message-wiadomość). System portów używa tych plików w specjalnych sytuacjach. Jeśli potrzebujesz więcej informacji na temat tych plików i portów w ogóle, zajrzyj do podręcznika FreeBSD Porter's Handbook (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/books/porters-handbook/index.html).

Jak już raz powiedziano, porty zawierają instrukcje odnośnie kompilacji programów z kodu źródłowego. Jednakże nie zawierają one samego kodu. Kod pobrać można z płyty CD bądź z Internetu. Rozprowadzany może być w dowolnej postaci jaką wybierze sobie jego producent, przy czym najczęściej jest to spakowany plik `tar` skompresowany dodatkowo `gzipem`. Kod źródłowy programu nazywany jest „`distfile`”. Poniżej przedstawione zostały dwie metody instalacji portów we FreeBSD.

Notatka: By móc zainstalować port musimy być zalogowania jako użytkownik `root`.

Ostrzeżenie Przed instalacją jakiegokolwiek portu należy upewnić się, że dysponujemy aktualną kolekcją portów oraz sprawdzić potencjalne luki bezpieczeństwa związane z danym portem na stronie <http://vuxml.freebsd.org/>.

Istnieje możliwość zautomatyzowania procesu weryfikacji potencjalnych luk bezpieczeństwa przed instalacją portu. Do tego celu można wykorzystać program **portaudit**, dostępny również w kolekcji portów (`security/portaudit`). Wydanie polecenia `portaudit -F` przed instalacją nowego portu spowoduje pobranie aktualnej bazy luk bezpieczeństwa. Możliwe jest również wykonywanie regularnych aktualizacji bazy i rewizji zainstalowanego oprogramowania w trakcie codziennego przeglądu bezpieczeństwa systemu. Więcej informacji dostępnych jest na stronach podręcznika systemowego `portaudit(1)` i `periodic(8)`.

Sposób funkcjonowania kolekcji portów wiąże się z założeniem, że posiadamy połączenie z Internetem. Jeśli nie, będziemy musieli ręcznie pobierać kod źródłowy „`distfile`” i umieszczać w katalogu `/usr/ports/distfiles` dla każdego instalowanego portu.

By rozpocząć instalację należy przejść do katalogu wybranego portu:

```
# cd /usr/ports/sysutils/lsof
```

Wewnątrz katalogu `lsof` znajduje się szkielet portu. Następnym krokiem jest kompilacja programu, co sprowadza się do wpisania polecenia `make`. Efekt działania polecenia powinien być zbliżony do:

```
# make
```

```
>> lsof_4.57D.freebsd.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
>> Attempting to fetch from ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/.
==> Extracting for lsof-4.57
...
[extraction output snipped]
...
>> Checksum OK for lsof_4.57D.freebsd.tar.gz.
==> Patching for lsof-4.57
==> Applying FreeBSD patches for lsof-4.57
==> Configuring for lsof-4.57
...
[configure output snipped]
...
==> Building for lsof-4.57
...
[compilation output snipped]
...
#
```

Po skończeniu kompilacji powracamy do linii poleceń. Kolejnym krokiem jest instalacja portu poprzez wpisanie polecenia `make` wraz ze słowem `install`:

```
# make install
==> Installing for lsof-4.57
...
[installation output snipped]
...
==> Generating temporary packing list
==> Compressing manual pages for lsof-4.57
==> Registering installation for lsof-4.57
==> SECURITY NOTE:
      This port has installed the following binaries which execute with
      increased privileges.
#
```

Gdy ponownie powrócimy do linii poleceń, powinniśmy być już w stanie uruchomić właśnie zainstalowaną aplikację. Ostrzeżenie jakie pojawi się na ekranie związane jest z faktem, że `lsof` jest programem pracującym ze zwiększonymi przywilejami. W trakcie kompilacji i instalacji portów powinniśmy zwracać uwagę na wszystkie pojawiające się ostrzeżenia.

Dobrym pomysłem, jest również usunięcie podkatalogu zawierającego wszystkie tymczasowe pliki wykorzystywane w trakcie kompilacji. Nie tylko dlatego, że niepotrzebnie zajmuje miejsce na dysku, ale również dlatego, że może być przyczyną problemów podczas aktualizacji programu do nowszej wersji.

```
# make clean
==> Cleaning for lsof-4.57
#
```

Notatka: Można sobie oszczędzić dwóch naddatkowych kroków wpisując od razu `make install clean` zamiast trzech osobnych poleceń `make`, `make install` oraz `make clean`.

Notatka: Niektóre powłoki utrzymują bufor listy poleceń z katalogów znajdujących się w zmiennej środowiskowej `PATH`. Ma to za zadanie przyspieszyć wyszukiwanie plików binarnych tychże poleceń. Jeśli korzystamy z jednej z takich właśnie powłok może okazać się niezbędnym wydać polecenie `rehash` po instalacji portu, nim będziemy mogli wykorzystać nowo zainstalowany program. Polecenie to dostępne jest przy wykorzystaniu powłoki typu `tcsh`. Natomiast dla powłoki typu `sh` odpowiednikiem jest `hash -r`. Więcej informacji dostępnych jest w dokumentacji powłoki.

Niektóre wydawnictwa na płytach DVD-ROM, jak np. FreeBSD Toolkit z FreeBSD Mall (<http://www.freebsdmail.com/>), zawierają źródła `distfile`. Mogą być one wykorzystane z kolekcją portów. Wystarczy zamontować płytę DVD w `/cdrom`. Jeśli natomiast używamy innego punktu montowania dla płyt musimy zmodyfikować zmienną `CD_MOUNTPTS` by wskazywała na właściwe miejsce. Niezbędne źródła `distfile` zostaną automatycznie wykorzystane jeśli znajdują się na płycie.

Notatka: Mimo wszystko należy mieć w pamięci, że licencje nielicznych portów nie zezwalają na załączenie ich na płycie CD-ROM. Może to być np. z powodu konieczności wcześniejszej rejestracji przed pobraniem źródeł bądź ich redystrybucja nie jest dozwolona. Jeśli chcemy zainstalować port, który nie znajduje się na płycie CD musimy mieć połączenie z Internetem.

System portów do pobierania plików wykorzystuje program `fetch(1)`, który z kolei potrafi korzystać z wielu zmiennych środowiskowych, m.in. `FTP_PASSIVE_MODE`, `FTP_PROXY` czy `FTP_PASSWORD`. Jeśli znajdujemy się za zaporą ogniową, bądź musimy korzystać z serwera pośredniczącego FTP/HTTP proxy, może się okazać, że będziemy musieli ustawić niektóre z tych zmiennych. Kompletna lista wykorzystywanych zmiennych dostępna jest w podręczniku systemowym `fetch(3)`.

Dla użytkowników nie mogących być cały czas połączonych z Internetem dostępne jest polecenie `make fetch`. Wystarczy wpisać to polecenie znajdując się w głównym katalogu drzewa portów (`/usr/ports`) a wymagane pliki zostaną automatycznie pobrane. Polecenie to będzie również funkcjonować w podkatalogach, np. `/usr/ports/net`. Jednakże, w takiej sytuacji *nie* zostaną automatycznie pobrane źródła bibliotek, od których zależy dany port. Zamieniając parametr `fetch` na `fetch-recursive` spowodujemy pobranie również źródeł wszystkich portów, od których zależy instalowany program.

Notatka: Możliwe jest kompilowanie każdego portu z osobna w danej kategorii, bądź wszystkich na raz poprzez polecenie `make` wykonane, analogicznie do `make fetch`, w głównym katalogu kategorii. Jednakże jest to niebezpieczna metoda, gdyż niektóre porty nie mogą jednocześnie funkcjonować w systemie, bądź mogą zainstalować różne pliki o tej samej nazwie.

W naprawdę żadkich przypadkach, użytkownicy mogą pozyskać pliki `distfile` z innego źródła niż `MASTER_SITES` (miejsce skąd je pobiera system portów). Opcję `MASTER_SITES` można zastąpić za pomocą następującego polecenia:

```
# cd /usr/ports/directory
# make MASTER_SITE_OVERRIDE= \
ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ fetch
```

W tym przykładzie zastąpiliśmy opcję `MASTER_SITES` adresem `ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/`.

Notatka: Niektóre porty umożliwiają (a nawet wymagają) podanie pewnych opcji kompilacji, które mogą włączyć bądź wyłączyć nie potrzebne części aplikacji, pewne opcje bezpieczeństwa i inne parametry. Z przychodzących na myśl tego typu programów to `www/mozilla`, `security/gpgme` oraz `mail/sylpheed-claws`. Za każdym razem gdy dostępne będą tego typu opcje wyświetlony zostanie komunikat.

4.5.2.1. Ignorowanie domyślnych katalogów portów

Czasami okazuje się być przydatne (a nawet wymagane) by skorzystać z innych katalogów tymczasowych i docelowych. Domyślne katalogi można zastąpić wykorzystując zmienne `WRKDIRPREFIX` i `PREFIX`. Na przykład:

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

spowoduje skompilowanie portu w katalogu `/usr/home/example/ports` i instalację w podkatalogach `/usr/local`.

```
# make PREFIX=/usr/home/example/local install
```

spowoduje natomiast kompilację w katalogu `/usr/ports` oraz instalację w podkatalogach `/usr/home/example/local`.

I oczywiście,

```
# make WRKDIRPREFIX=../ports PREFIX=../local install
```

spowoduje połączenie obydwu powyższych ustawień (jest to za długie by całkowicie zmieściło się na stronie, ale powinno dać ogólne wyobrażenie).

Alternatywnie, obydwa zmienne mogą być również określone jako zmienne środowiskowe. Informacje o definiowaniu zmiennych środowiskowych dostępne są w podręczniku systemowym naszej powłoki.

4.5.2.2. Jak poradzić sobie z `imake`

Niektóre porty wykorzystujące `imake` (część Systemu okien X) nie współpracują ze zmienną `PREFIX` i mimo wszystko będą instalowały programy w `/usr/X11R6`. Podobnie niektóre z portów napisanych w języku Perl ignorują zmienną `PREFIX` i instalują programy w głównym drzewie Perla. Zmuszenie tych portów do współpracy ze zmienną `PREFIX` jest niezmiernie trudne, albo wręcz niemożliwe.

4.5.3. Usuwanie zainstalowanych portów

Teraz, gdy wiesz już jak instalować porty, zastanawiasz się prawdopodobnie jak je usuwać, na przykład w wypadku, gdy zainstalowaliśmy port, ale okazało się jednak, że to nie był ten którego szukaliśmy. W ramach przykładu usuniemy port, który instalowaliśmy poprzednio (dla tych którzy nie uważają, był to `lsOf`). Podobnie jak w przypadku pakietów (szerzej opisane w podrozdziale traktującym o pakietach), również porty usuwane są za pomocą polecenia `pkg_delete(1)`:

```
# pkg_delete lsOf-4.57
```

4.5.4. Aktualizacja portów

Na wstępie musimy wyświetlić zdezaktualizowane porty w kolekcji. Wykorzystamy do tego polecenie `pkg_version(1)`:

```
# pkg_version -v
```

4.5.4.1. /usr/ports/UPDATING

Po zaktualizowaniu kolekcji, a przed próbą aktualizacji jakichkolwiek portów, należy zapoznać się z zawartością pliku `/usr/ports/UPDATING`. Plik ten opisuje różne zagadnienia i dodatkowe kroki, na które można natknąć się i będzie trzeba wykonać podczas aktualizacji, np. zmiany formatu plików czy zmiany w lokalizacji plików konfiguracyjnych.

Jeśli opis w pliku `UPDATING` mówi coś innego niż ten tekst, należy zastosować się do opisu.

4.5.4.2. Aktualizacja portów z wykorzystaniem programu Portupgrade

Program **portupgrade** został zaprojektowany by ułatwić aktualizację zainstalowanych w systemie portów. Dostępny jest z portu `sysutils/portupgrade`. Jego instalacja przebiega dokładnie tak samo, jak każdego innego portu, wykorzystując polecenie `make install clean` command:

```
# cd /usr/ports/sysutils/portupgrade
# make install clean
```

Przeskanujmy następnie listę zainstalowanych portów za pomocą polecenia `pkgdb -F` i usuńmy wszystkie niezgodności jakie nam zwróci skanowanie. Regularne skanowanie przed każdą aktualizacją jest zdecydowanie dobrym pomysłem.

Wydanie polecenia `portupgrade -a` spowoduje, że program **portupgrade** rozpocznie aktualizację wszystkich przedawnionych portów zainstalowanych w naszym systemie. Parametr `-i` pozwoli przejść w tryb interaktywny, gdzie będziemy musieli potwierdzić aktualizację każdego portu.

```
# portupgrade -ai
```

By zaktualizować jedynie wybraną aplikację zamiast wszystkich portów należy wykorzystać polecenie `portupgrade nazwa_programu`. Opcja `-R` oznacza, że `portupgrade` powinien najpierw zaktualizować wszystkie porty, od których zależy dany program.

```
# portupgrade -R firefox
```

By do instalacji wykorzystać pakiety zamiast portów należy dodać parametr `-P`. Wówczas **portupgrade** przeszuka katalogi zawarte w zmiennej `PKG_PATH`. Jeśli pakiet nie zostanie odnaleziony lokalnie zostanie pobrany z Internetu. Jeśli nie będzie możliwe żadne z powyższych, wówczas **portupgrade** wykorzysta do aktualizacji porty. By temu zapobiec należy zastosować parametr `-PP`.

```
# portupgrade -PR gnome2
```

Aby pobrać jedynie pliki źródłowe `distfiles` (bądź pakiety, gdy wykorzystamy opcję `-P`) bez kompilacji czy instalacji czegokolwiek należy użyć parametru `-F`. Więcej informacji dostępnych jest w `portupgrade(1)`.

4.5.4.3. Aktualizacja portów z wykorzystaniem programu Portmanager

Kolejnym narzędziem ułatwiającym aktualizację zainstalowanych portów jest **Portmanager**, dostępny z portu `sysutils/portmanager`:

```
# cd /usr/ports/sysutils/portmanager
# make install clean
```

Wszystkie zainstalowane porty mogą zostać zaktualizowane za pomocą polecenia:

```
# portmanager -u
```

Wykorzystując parametr `-ui` przechodzimy w tryb interaktywny, gdzie będziemy pytani o potwierdzenie każdej operacji wykonywanej przez **Portmanager**. Program ten może być z równym powodzeniem wykorzystywany do instalacji nowych portów w systemie. W przeciwieństwie do polecenia `make install clean` program **Portmanager** zaktualizuje wszystkie zależności nim skompiluje i zainstaluje wybrany port.

```
# portmanager x11/gnome2
```

Gdy wystąpią problemy z zależnościami wybranego portu można wykorzystać **Portmanagera**, by ponownie skompilował je we właściwej kolejności. Na koniec zostanie również ponownie skompilowany port stwarzający problemy.

```
# portmanager graphics/gimp -f
```

Więcej informacji dostępnych jest na stronach podręcznika systemowego **Portmanagera**.

4.5.5. Porty i przestrzeń na dysku

Korzystanie z kolekcji portów z czasem odbija się na wolnym miejscu na dysku. Dlatego też zawsze po skompilowaniu i zainstalowaniu programu z portu powinniśmy pamiętać o usunięciu tymczasowych katalogów roboczych (ang. *work directories*) wykorzystując do tego polecenie `make clean`. Całą kolekcję natomiast można oczyścić wpisując polecenie:

```
# portsclean -C
```

Z czasem uzbiera nam się wiele katalogów `distfiles`, które będą jedynie zajmować przestrzeń na dysku. Możemy je ręcznie usuwać bądź posłużyć się następującym poleceniem, by usunąć wszystkie katalogi `distfiles` nie powiązane aktualnie z żadnym portem:

```
# portsclean -D
```

Badź, by usunąć wszystkie katalogi `distfiles`, do których nie odnosi się żaden z aktualnie zainstalowanych portów w naszym systemie:

```
# portsclean -DD
```

Notatka: Program `portsclean` jest częścią pakietu **portupgrade**.

Pamiętajmy również o usuwaniu instalowanych portów gdy już ich nie potrzebujemy. Przydatne narzędzie pozwalające zautomatyzować te czynności znajduje się w `sysutils/pkg_cutleaves`.

4.6. Czynności po-instalacyjne

Po zainstalowaniu nowego programu z reguły chcemy zapoznać się z dostarczoną z nim dokumentacją, zmodyfikować wymagane pliki konfiguracyjne, upewnić się, że program (jeśli jest to demon) będzie uruchamiany w trakcie ładowania systemu, itp.

Oczywiście, szczegółowe kroki jakie należy podjąć konfigurując każdą aplikację będą różne. Tym nie mniej, jeśli właśnie zainstalowaliśmy nowy program i zastanawiamy się „Co dalej?” poniższe uwagi mogą okazać się pomocne:

- Za pomocą `pkg_info(1)` możemy sprawdzić gdzie i jakie pliki zostały zainstalowane. Na przykład, jeśli zainstalowaliśmy wersję 1.0.0 pakietu `FooPackage`, polecenie

```
# pkg_info -l foopackage-1.0.0 | less
```

wyświetli nam wszystkie pliki zainstalowane z pakietu. Szczególną uwagę warto zwrócić na pliki zainstalowane w katalogach: `man/` zawierającym strony podręcznika systemowego, `etc/` zawierającym pliki konfiguracyjne, oraz `doc/`, gdzie znajdować się będzie dużo obszerniejsza dokumentacja.

Jeśli nie jesteśmy pewni, którą wersją programu zainstalowaliśmy, polecenie

```
# pkg_info | grep -i foopackage
```

wyświetli wszystkie zainstalowane pakiety zawierające `foopackage` w nazwie. Oczywiście `foopackage` należy zastąpić nazwą poszukiwanego pakietu.

- Gdy już udało się ustalić jakie strony podręcznika systemowego zostały zainstalowane przez dany pakiet, można je przeczytać za pomocą polecenia `man(1)`. Warto również obejrzeć przykładowe pliki konfiguracyjne i wszelką dodatkową dokumentację.
- Jeśli dana aplikacja posiada własną witrynę internetową warto jest również tam poszukać dodatkowej dokumentacji czy odpowiedzi na często zadawane pytania (FAQ). Jeśli nie znamy właściwego adresu internetowego może być on podany w wyniku polecenia

```
# pkg_info foopackage-1.0.0
```

Wiersz `WWW:`, jeśli w ogóle jest podany, powinien zawierać informacje o adresie witryny.

- Programy, które powinny być uruchamiane podczas ładowania systemu (np. serwery internetowe) z reguły instalują przykładowy skrypt w `/usr/local/etc/rc.d`. Powinniśmy sprawdzić zawartość tego skryptu oraz razie potrzeby zmodyfikować go bądź zmienić nazwę. Szczegółowe informacje dostępne są w podrozdziale Uruchamianie usług.

4.7. Jak radzić sobie ze źle przygotowanymi portami

Jeśli natknęliśmy się na port, który z jakichś powodów nie działa na naszym komputerze, możemy zrobić kilka następujących rzeczy:

1. Sprawdzić w bazie danych zgłoszonych problemów (<http://www.FreeBSD.org/support.html#gnats>) czy jest przygotowywana poprawka dla danego portu. Jeśli tak, może uda się nam zastosować tę poprawkę.

2. Poprosić o pomoc opiekuna danego portu. Adres email opiekuna można znaleźć przeglądając plik `Makefile` w katalogu portu bądź wpisując polecenie `make maintainer`. Wysyłając wiadomość pamiętajmy o zawarciu informacji o nazwie i wersji portu (najlepiej jest zawrzeć cały wiersz z pliku `Makefile` zaczynający się od `$FreeBSD:`), oraz opis błędu i wynik działania programu w momencie zaistnienia błędu.

Notatka: Niektóre porty nie są przygotowywane przez pojedyncze osoby, ale raczej przez grupy dyskusyjne (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/mailling-list-faq/article.html). Wiele adresów takich grup, choć nie wszystkie, ma postać `<freebsd-listname@FreeBSD.org>`. Należy mieć również to na uwadze formułując swoje pytania.

Porty przygotowywane przez `<freebsd-ports@FreeBSD.org>` w rzeczywistości nie posiadają żadnego konkretnego opiekuna, ani grupy opiekunów. Poprawki i pomoc dla takich portów przygotowują osoby zapisane na tę listę dyskusyjną. Nowi ochotnicy są zawsze mile widziani!

W przypadku braku odpowiedzi można również przesłać zgłoszenie błędu poprzez `send-pr(1)` (szczegóły w artykule *Writing FreeBSD Problem Reports* (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/problem-reports/article.html)).

3. Naprawić błąd samemu! Podręcznik *Porter's Handbook* (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/books/porters-handbook/index.html) (ang.) zawiera szczegółowe informacje o strukturze „Portów”, dzięki czemu można samemu naprawić błąd lub przygotować własny port!
4. Pobrać pakiet z najbliższego serwera FTP. „Główne” repozytorium pakietów znajduje się na serwerze `ftp.FreeBSD.org` w katalogu `packages` (<ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/>). Tym nie mniej warto jest najpierw odszukać lokalny serwer lustrzany (<http://mirrorlist.FreeBSD.org/>). Szanse na to, że gotowe pakiety będą działać poprawnie są większe niż w przypadku kompilowania programów. Pakiety można zainstalować za pomocą programu `pkg_add(1)`.

Rozdział 5. System okien X

Uzupełnili o serwer X.Org Ken Tom i Marc Fonvieille. Tłumaczył Cezary Morgia.

5.1. Streszczenie

Środowisko graficzne dostępne we FreeBSD korzysta z zaawansowanego serwera graficznego X11 — implementacji open-source Systemu okien X obejmującej zarówno **Xorg** jak i **XFree86**. FreeBSD 5.2.1-RELEASE oraz wcześniejsze wydania wykorzystują **XFree86**, serwer X11 opracowany przez The XFree86 Project, Inc. Od wersji FreeBSD 5.3-RELEASE podstawową i oficjalną odmianą X11 jest **Xorg**, serwer przygotowywany przez X.Org Foundation, rozprawdzany na licencji bardzo zbliżonej do wykorzystywanej przez FreeBSD. Dostępne są również komercyjne serwery X dla FreeBSD.

Niniejszy rozdział omawia zagadnienia związane z instalacją i konfiguracją X11 kładąc szczególny nacisk na serwer **Xorg**. Informacje o konfiguracji **XFree86** (np. w starszych wersjach FreeBSD gdzie **XFree86** był domyślnym serwerem X11) zawsze znaleźć można w archiwalnych wersjach Podręcznika FreeBSD (ang.) na stronie <http://docs.FreeBSD.org/doc/>.

Informacje odnośnie obsługiwanego przez X11 sprzętu dostępne są na stronie internetowej projektu Xorg (<http://www.x.org/>).

Po przeczytaniu tego rozdziału będziemy wiedzieć:

- Jakie elementy wchodzą w skład Systemu okien X i jakie są ich wzajemne relacje.
- Jak zainstalować i skonfigurować X11.
- Jak instalować i korzystać z różnych menadżerów okien.
- Jak korzystać z czcionek TrueType® w X11.
- Jak skonfigurować system do logowania graficznego (**XDM**).

Przed przeczytaniem tego rozdziału powinniśmy wiedzieć:

- Jak instalować dodatkowe oprogramowanie (Rozdział 4).

5.2. Zrozumieć X

Korzystanie z X pierwszy raz może być niejakim szokiem dla osób, które dotychczas korzystały z innych środowisk graficznych, jak np. Microsoft Windows czy Mac OS.

O ile nie jest wymagane znać wszystkie detale wielu elementów X i jak one ze sobą współpracują, o tyle podstawowa wiedza w tym zakresie pozwoli nam w pełni wykorzystać możliwości X-ów.

5.2.1. Czemu X?

X nie jest pierwszym systemem okienkowym napisanym dla systemów typu UNIX, lecz jest on najbardziej popularnym. Grupa projektantów, która przygotowała X, pracowała wcześniej nad innym systemem. System ten nazywał się „W” (od „Window”). X była po prostu kolejną literą w rzymskim alfabecie.

System X może być nazywany po prostu „X”, „System okien X”, „X11” oraz jeszcze na wiele innych sposobów. Może się również okazać, że stosowanie terminu „X Windows” w odniesieniu do X11 jest traktowane jako obraźliwe przez niektóre osoby. Więcej informacji dostępnych jest w X(7).

5.2.2. Model klient/serwer

Od samego początku System X zorientowany był na pracę w sieci, stąd też wykorzystanie modelu „klient-serwer”.

W modelu systemu X, „serwer X” pracuje na komputerze wyposażonym w klawiaturę, monitor i myszkę. Do zadań serwera należy m.in. zarządzanie wyświetlaniem, czy obsługa sygnałów z klawiatury. Każda aplikacja graficzna (jak np. **XTerm** czy **Netscape**) jest „klientem”. Klient wysyła komunikaty do serwera typu „Proszę w tym miejscu narysować okienko”. Serwer natomiast: „Użytkownik właśnie kliknął przycisk OK”.

W warunkach domowych czy w małym biurze serwer i klienci pracują z reguły na tym samym komputerze. Tym nie mniej istnieje możliwość uruchomienia serwera X na słabszej maszynie a aplikacje (klienci) na np. potężnej i drogiej maszynie obsługującej całe biuro. W takim wypadku komunikacja pomiędzy klientami a serwerem odbywa się za pomocą sieci.

Bywa to mylące, gdyż terminologia stosowana w systemie X jest dokładnie odwrotna do tego czego należałoby się spodziewać w typowym modelu „klient-serwer”, czyli „serwera X” pracującego na mocniejszej maszynie oraz „klienta X” na komputerze biurowym.

Stąd też należy pamiętać, że serwer X jest komputerem z monitorem i klawiaturą, podczas gdy klienci X są programami wyświetlającymi okienka.

Protokół X11 w żaden sposób nie zmusza ani klientów ani serwera, by obydwa działały na tym samym systemie operacyjnym, czy nawet typie komputera. Możliwe jest zatem uruchomienie serwera X w systemie Microsoft Windows czy Mac OS firmy Apple za pomocą dostępnych darmowych i komercyjnych narzędzi.

5.2.3. Menedżer okien

Filozofia systemu X jest bardzo zbliżona do filozofii Uniksa: „narzędzia, nie reguły”. Oznacza to, że X nie stara się narzucać jak ma zostać wykonane zadanie, dostarcza jedynie narzędzi pozostawiając użytkownikowi decyzję o sposobie ich wykorzystania.

Stąd też X nie wymusza jak powinny wyglądać okienka, jak je przesuwać po ekranie za pomocą myszki, jakie skróty klawiaturowe wykorzystać by przełączać pomiędzy okienkami (np. **Alt+Tab** w w przypadku Microsoft Windows), jak powinny wyglądać nagłówki okienek, itd.

Serwer X oddelegowuje tę odpowiedzialność do aplikacji nazywanej „Menedżerem okien”. Istnieje całe mnóstwo menedżerów okien dla systemu X: **AfterStep**, **Blackbox**, **ctwm**, **Enlightenment**, **fvwm**, **Sawfish**, **twm**, **Window Maker** i wiele więcej. Każdy z nich oferuje inny wygląd i sposób obsługi; niektóre obsługują „wirtualne pulpity”; inne umożliwiają definiować własne skróty klawiszowych do zarządzania pulpitem; jeszcze inne posiadają przycisk „Start” bądź podobne rozwiązanie; w niektórych można zmieniać dowolnie motywy graficzne, pozwalając na całkowitą zmianę wyglądu i zachowania przy uruchamianiu nowego motywu. Te menedżery i wiele innych dostępne są w kategorii **x11-wm** w Kolekcji portów.

Ponadto, również środowiska graficzne **KDE** oraz **GNOME** posiadają w pełni zintegrowane, własne menedżery okien.

Każdy menedżer okien posiada również odrębny mechanizm konfiguracyjny; niektóre wykorzystują ręcznie modyfikowane pliki konfiguracyjne, inne dysponują narzędziami graficznymi, a przynajmniej jeden (**Sawfish**) wykorzystuje plik konfiguracyjny napisany w języku Lisp.

Sposób uaktywniania: Kolejną funkcją realizowaną przez menedżera okien jest „sposób uaktywniania” okien za pomocą myszy. Każdy system okienkowy potrzebuje pewnego sposobu wyboru okna, które będzie aktywnie przyjmować sygnały z klawiatury i powinno wskazywać, które okno jest aktywne.

Znanym wszystkim sposobem uaktywniania jest zapewne „kliknij-by-uaktywnić”. Jest to metoda wykorzystywana w systemie Microsoft Windows, w której okno zostaje uaktywnione po otrzymaniu kliknięcia myszką.

X nie obsługuje żadnego sposobu uaktywniania sam z siebie. To właśnie menedżer okien kontroluje, które okno jest aktywne w danych czasie. Różne menedżery wspierają różne metody uaktywniania. Wszystkie z nich obsługują kliknij-by-uaktywnić a większość z nich obsługuje również kilka innych.

Najczęściej spotykane sposoby uaktywniania:

aktywuj-za-myszą

Aktywne jest okno znajdujące się bezpośrednio pod wskaźnikiem myszki. Przy czym, nie koniecznie jest to te samo okno, które znajduje się nad wszystkimi innymi oknami. Zmiana aktywnego okna dokonywana jest przez wskazanie na inne okno. Nie jest wymagane kliknięcie na nim.

leniwe uaktywnianie

Ta metoda jest drobną wariacją metody aktywuj-za-myszą, w której w sytuacji gdy wskaźnik myszy jest przesunięty nad wolne pole wówczas żadne okno nie jest aktywne, a wszystkie wprowadzane znaki są tracone. W tej metodzie natomiast aktywne okno jest zmieniane tylko gdy wskaźnik zostanie przesunięty nad nowe okno, natomiast nie w momencie gdy opuści bieżące okno.

kliknij-by-uaktywnić

Aktywne okno jest wybierane poprzez kliknięcie na nie myszką. Okno może później być „podniesione” i pojawić się nad wszystkimi innymi oknami. Wszystkie wprowadzane znaki są kierowane do tego okna, nawet jeśli wskaźnik myszki zostanie przesunięty nad inne okno.

Wiele menedżerów okien wspiera również inne metody, podobnie jak wariacje powyższych. Najlepiej jest sprawdzić dokumentację danego menedżera.

5.2.4. Elementy interfejsu graficznego

Podejście Systemu X do dostarczania narzędzi a nie reguł dotyczy również elementów interfejsu graficznego widocznych na ekranie w każdej uruchomionej aplikacji.

Pod pojęciem „elementu interfejsu graficznego” (ang. widget) kryją się wszystkie elementy, które można kliknąć bądź w inny sposób nimi manipulować; przyciski, pola wyboru, przyciski opcji, ikony, listy, itd. W systemach Microsoft Windows nazywają się one „formantkami” (ang. controls).

Zarówno Microsoft Windows jak i Apple Mac OS stosują bardzo rygorystyczne podejście do elementów interfejsu graficznego. Od twórców programów wymaga się by ich aplikacje wyglądały jednakowo. Natomiast przy tworzeniu X, nie uznano za rozsądne narzucanie jednego stylu graficznego czy zestawu elementów interfejsu, do którego miałyby być dostosowane wszystkie programy.

W rezultacie nie należy się spodziewać, że aplikacje graficzne będą posiadały jednakowy wygląd czy sposób obsługiwanie. Istnieje kilka popularnych zestawów elementów interfejsu graficznego i ich wariacji, włączając w to oryginalny zestaw Athena z MIT, **Motif®** (na podstawie którego został przygotowany zestaw elementów interfejsu graficznego Microsoft Windows; wszystkie krawędzie fazowane, trzy odcienie szarości), **OpenLook** i inne.

Większość nowszych programów graficznych będzie zapewne wykorzystywać jeden ze współczesnych zestawów elementów interfejsu, jak np. Qt, wykorzystywany w **KDE**, bądź GTK+, stosowany w projekcie **GNOME**. Pod tym względem, istnieje pewne podobieństwo w wyglądzie i zachowaniu środowisk graficznych w systemach typu UNIX, co z pewnością ułatwi pracę z systemem początkującym użytkownikom.

5.3. Instalacja X11

Domyślną implementacją serwera X11 dla FreeBSD jest **Xorg**. Jest on serwerem graficznym o otwartym kodzie (open source) Systemu okien X przygotowanym przez X.Org Foundation. Bazuje on na kodzie **XFree86 4.4RC2** i X11R6.6. Wersją **Xorg** dostępną aktualnie z kolekcji portów FreeBSD jest 7.5.2.

By skompilować i zainstalować **Xorg** z kolekcji portów wpisujemy:

```
# cd /usr/ports/x11/xorg
# make install clean
```

Notatka: Nim zaczniemy upewnijmy się, że dysponujemy 4 GB wolnej przestrzeni na dysku na potrzeby kompilacji.

Alternatywnie, X11 może zostać zainstalowany z pakietów binarnych za pomocą `pkg_add(1)`. W przypadku wykorzystania opcji zdalnego pobierania pakietów z sieci przez `pkg_add(1)` należy pominąć numer wersji pakietu. Program `pkg_add(1)` automatycznie pobierze najnowszą wersję aplikacji.

Zatem, by pobrać i zainstalować pakiet **Xorg** wystarczy wpisać:

```
# pkg_add -r xorg
```

Notatka: Powyższe polecenia zainstalują kompletne środowisko X11 zawierające serwery, klienty, czcionki itd. Dostępne są również osobne pakiety i porty elementów X11.

Pozostała część niniejszego rozdziału wyjaśnia jak skonfigurować serwer X11 oraz jak skonfigurować wspomagające efektywność naszej pracy środowisko pulpitu.

5.4. Konfiguracja X11

Napisał Christopher Shumway.

5.4.1. Nim zaczniemy

Przed rozpoczęciem konfiguracji X11 potrzebne nam będą następujące informacje o docelowym systemie:

- Parametry monitora
- Rodzaj chipsetu karty graficznej
- Rozmiar pamięci karty graficznej

Parametry monitora są wykorzystywane przez X11 do określenia rozdzielczości i częstotliwości odświeżania ekranu, na jakich ma pracować. Parametry te można z reguły odczytać z dokumentacji dostarczonej wraz z monitorem bądź ze strony producenta. Potrzebne są dwa przedziały liczbowe: częstotliwość odchyłania poziomego oraz częstotliwość synchronizacji pionowej.

Od rodzaju chipsetu karty graficznej zależy który moduł X11 wykorzysta do komunikacji z kartą graficzną. W większości przypadków możliwe jest automatyczne wykrycie rodzaju chipsetu. Tym nie mniej warto jest go znać, na wypadek gdyby autodetekcja nie powiodła się.

Rozmiar pamięci karty graficznej wpływa bezpośrednio na rozdzielczość i głębię kolorów, przy których system będzie pracował. Informacja ta jest istotna, by użytkownik znał ograniczenia systemu w tym zakresie.

5.4.2. Konfiguracja X11

Konfiguracja X11 jest procesem składającym się z kilku kroków. Pierwszym z nich jest przygotowanie wstępnego pliku konfiguracyjnego. Wystarczy jako użytkownik `root` wpisać:

```
# Xorg -configure
```

Wygeneruje to szkielet konfiguracji X11 w katalogu `/root`, w pliku o nazwie `xorg.conf.new` (niezależnie czy skorzystaliśmy z `su(1)` czy zalogowaliśmy się bezpośrednio na konto, plik zostanie utworzony w katalogu zdefiniowanym w zmiennej `$HOME` dla użytkownika `root`). X11 spróbuje wykryć parametry sprzętu graficznego zainstalowanego w komputerze i zapisać plik konfiguracyjny, by przy starcie serwera X były ładowane właściwe sterowniki dla wykrytego sprzętu.

Kolejnym krokiem jest przetestowanie konfiguracji i sprawdzenie czy **Xorg** jest w stanie współpracować ze sprzętem graficznym w systemie. W tym celu należy wpisać:

```
# Xorg -config xorg.conf.new
```

Jeśli na ekranie pojawi się siatka złożona z czarnych i szarych elementów, a także kursor myszy w kształcie litery X, oznaczać to będzie, że X11 został skonfigurowany poprawnie. By wyłączyć ekran testowy wystarczy wcisnąć kombinację klawiszy **Ctrl+Alt+Backspace**.

Notatka: Jeśli okaże się, że kursor nie reaguje na ruchy myszy będziemy musieli wpierw ją skonfigurować. Sekcja 2.9.10 rozdziału „Instalacja FreeBSD” zawiera szczegółowe informacje na ten temat.

Następnym krokiem jest dostrojenie konfiguracji pliku `xorg.conf.new` do naszych upodobań. Otwórzmy plik w edytorze tekstu, np. w `emacs(1)` bądź `ee(1)`. Wpierw powinniśmy dodać częstotliwości z jakimi może pracować nasz monitor. Z reguły określane są jako częstotliwości synchronizacji pionowej i poziomej. Wartości te są dodawane do pliku `xorg.conf.new` w sekcji "Monitor":

```
Section "Monitor"
    Identifier      "Monitor0"
    VendorName      "Monitor Vendor"
```

```

    ModelName      "Monitor Model"
    HorizSync      30-107
    VertRefresh    48-120
EndSection

```

Słów kluczowych `HorizSync` i `VertRefresh` może brakować w pliku konfiguracyjnym. Jeśli tak jest, można je śmiało dodać wpisując właściwą wartość częstotliwości odchyłania poziomego zaraz po `HorizSync` oraz wartość częstotliwości synchronizacji pionowej po `VertRefresh`. W powyższym przykładzie wartości te zostały wpisane.

X umożliwia również korzystanie z funkcji DPMS (Energy Star), jeśli dysponujemy monitorem zgodnym z tym standardem. Program `xset(1)` kontroluje limity czasowe i może wymusić tryb oczekiwania, zawieszenia czy tryby wyłączania. Jeśli chcemy włączyć funkcje DPMS dla naszego monitora, musimy dodać poniższy wiersz w sekcji monitora:

```
Option      "DPMS"
```

Mając wciąż otwarty w edytorze plik `xorg.conf.new` wybierzmy domyślną rozdzielczość i głębię kolorów. Parametry te definiowane są w sekcji "Screen":

```

Section "Screen"
    Identifier "Screen0"
    Device     "Card0"
    Monitor    "Monitor0"
    DefaultDepth 24
    SubSection "Display"
        Viewport 0 0
        Depth    24
        Modes     "1024x768"
    EndSubSection
EndSection

```

Słowo kluczowe `DefaultDepth` odnosi się do domyślnej głębi kolorów. Opcja ta może być zmieniona za pomocą parametru `-depth` polecenia `Xorg(1)`. Słowo kluczowe `Modes` odnosi się do rozdzielczości, w której ma pracować serwer X dla danej głębi kolorów. Należy zwrócić uwagę, iż dostępne są jedynie standardowe tryby VESA, zgodne ze sprzętem graficznym instalowanym w danym systemie. W powyższym przykładzie, domyślna głębia kolorów to dwadzieścia cztery bity na piksel. Przy tej głębi dostępna jest rozdzielczość 1024 na 768 pikseli.

Możemy w końcu zapisać plik konfiguracyjny i sprawdzić go wykorzystując podany powyżej tryb testowy.

Notatka: Jednym z pomocnych narzędzi w radzeniu sobie z problemami są pliki dzienników X11, zawierające informacje o każdym urządzeniu, do którego jest podłączony serwer X11. Nazwy plików dzienników **Xorg** wykorzystują format `/var/log/Xorg.0.log`. Dokładna nazwa pliku dziennika może być różna w zakresie od `Xorg.0.log` do `Xorg.8.log`.

Jeśli test wypadł dobrze, należy zainstalować plik konfiguracyjny w miejscu gdzie `Xorg(1)` będzie w stanie go znaleźć. Z reguły jest to `/etc/X11/xorg.conf` lub `/usr/X11R6/etc/X11/xorg.conf`.

```
# cp xorg.conf.new /etc/X11/xorg.conf
```

Proces konfiguracji X11 dobiegł końca. **Xorg** można uruchomić za pomocą polecenia `startx(1)`. Serwer X11 może być również uruchamiany wykorzystując `xdm(1)`.

Notatka: Dostępny jest również graficzny konfigurator - `xorgcfg(1)` - dostarczany razem z dystrybucją X11. Pozwala on nam interaktywnie zdefiniować naszą konfigurację wybierając odpowiednie sterowniki i ustawienia. Program ten można uruchomić z konsoli wpisując polecenie `xorgcfg -textmode`. Więcej szczegółów zawiera strona podręcznika systemowego `xorgcfg(1)`.

Istnieje również, jako alternatywa, program `xorgconfig(1)`, będący aplikacją konsolową, mniej przyjazną dla początkujących użytkowników, jednakże przydatną w sytuacjach gdy inne narzędzia nie działają poprawnie.

5.4.3. Konfiguracja zaawansowana

5.4.3.1. Konfiguracja z chipsetem graficznym Intel® i810

Konfiguracja ze zintegrowanym chipsetem Intel i810 wymaga wykorzystania interfejsu programowego AGP `agpgart` do obsługi karty w X11. Więcej informacji można znaleźć w podręczniku systemowym sterownika `agp(4)`.

Pozwoli to nam skonfigurować naszą kartę graficzną jak każdą inną. W tym momencie należy zwrócić uwagę na fakt, iż w systemach bez `agp(4)` wkompiłowanego w jądro, próba załadowania modułu za pomocą `kldload(8)` nie powiedzie się. Sterownik ten musi być obecny w jądrze w trakcie uruchamiania systemu poprzez wkompiłowanie go bądź załadowanie za pomocą `/boot/loader.conf`.

5.4.3.2. Dodanie płaskiego monitora szerokokątnego

Sekcja ta zakłada, że posiadamy odrobinę wiedzy o zaawansowanej konfiguracji X11. Jeśli próby wykorzystania opisanych wyżej standardowych narzędzi konfiguracyjnych nie powiodły się, w plikach dzienników znajdziemy dostateczną ilość informacji pomocnych w uruchomieniu X z monitorem szerokokątnym. Będziemy musieli wykorzystać dowolny edytor tekstu.

Obecne formaty szerokokątne (WSXGA, WSXGA+, WUXGA, WXGA, WXGA+, itd.) obsługują formaty 16:10 oraz 10:9 bądź o innych proporcjach obrazu, które mogą stworzyć problemy w trakcie konfiguracji X. Niektórymi z powszechnie wykorzystywanych rozdzielczości ekranu dla proporcji 16:10 są:

- 2560x1600
- 1920x1200
- 1680x1050
- 1440x900
- 1280x800

W pewnym momencie będzie to tak proste jak dodanie którejś z tych rozdzielczości jako możliwych trybów `Mode` w `Section "Screen"`, jak np.:

```
Section "Screen"
Identifier "Screen0"
Device      "Card0"
Monitor     "Monitor0"
DefaultDepth 24
SubSection "Display"
    Viewport 0 0
```

```

        Depth      24
        Modes      "1680x1050"
EndSubSection
EndSection

```

Tym nie mniej **Xorg** jest na tyle sprytny, że potrafić pozyskać informacje o rozdzielczości ekranu monitora szerokokątnego za pomocą I2C/DDC w taki sposób, że wie jakie rozdzielczości potrafi obsłużyć monitor w kwestii częstotliwości i rozdzielczość.

Jeśli odpowiednie wpisy `ModeLine` nie istnieją w sterownikach, będziemy musieli podpowiedzieć co nieco serwerowi **Xorg**. Z pliku `/var/log/Xorg.0.log` możemy wydobyć dostateczną ilość informacji, by móc ręcznie stworzyć poprawnie obsługiwany `ModeLine`. Wystarczy odnaleźć zapis podobny do:

```

(II) MGA(0): Supported additional Video Mode:
(II) MGA(0): clock: 146.2 MHz   Image Size:  433 x 271 mm
(II) MGA(0): h_active: 1680   h_sync: 1784   h_sync_end 1960 h_blank_end 2240 h_border: 0
(II) MGA(0): v_active: 1050   v_sync: 1053   v_sync_end 1059 v_blanking: 1089 v_border: 0
(II) MGA(0): Ranges: V min: 48   V max: 85 Hz, H min: 30   H max: 94 kHz, PixClock max 170 MHz

```

Jest to tzw. informacja EDID. Stworzenie na jej podstawie `ModeLine` jest zaledwie kwestią wpisania we właściwej kolejności kilku liczb:

```
ModeLine <name> <clock> <4 horiz. timings> <4 vert. timings>
```

Tak więc wpis `ModeLine` w `Section "Monitor"` dla tego przykładu wyglądałby następująco:

```

Section "Monitor"
Identifier      "Monitor1"
VendorName     "Bigname"
ModelName      "BestModel"
ModeLine       "1680x1050" 146.2 1680 1784 1960 2240 1050 1053 1059 1089
Option         "DPMS"
EndSection

```

Po tych kilku prostych zmianach X powinien zacząć działać poprawnie z naszym szerokokątnym monitorem.

5.5. Korzystanie z czcionek w X11

Napisał Murray Stokely.

5.5.1. Czcionki Type1

Czcionki dostarczane razem z X11 są dalekie od idealnych dla typowych aplikacji biurowych. Duże czcionki sprawiają wrażenie postrzępionych i mało profesjonalnych. Natomiast, małe czcionki w **Netscape** są całkowicie nieczytelne. Tym nie mniej, dostępnych jest kilka darmowych, wysokiej jakości czcionek Type1 (PostScript®), gotowych do użycia z X11. Na przykład, kolekcja czcionek URW (`x11-fonts/urwfonts`) zawiera wysokiej jakości wersje standardowych czcionek type1 (Times Roman®, Helvetica®, Palatino® i innych). Kolekcja Freefonts (`x11-fonts/freefonts`) to wiele dodatkowych czcionek, przy czym większość z nich przewidzianych jest do użycia z oprogramowaniem graficznym, jak np. **Gimp**, tym samym nie są przygotowane do wykorzystania jako

czcionki do aplikacji. Dodatkowo, przy minimum wysiłku, można skonfigurować X11, by korzystał z czcionek TrueType. Więcej szczegółów znaleźć można w podręczniku systemowym X(7) lub w części poświęconej czcionkom TrueType.

By zainstalować kolekcje czcionek Type1 z portów, należy wpisać następujące polecenia:

```
# cd /usr/ports/x11-fonts/urwfonts
# make install clean
```

Analogicznie postępujemy z czcionkami freetype bądź innymi kolekcjami. Aby serwer X wykrył zainstalowane czcionki, należy dodać odpowiedni wpis do pliku konfiguracji serwera (/etc/X11/xorg.conf) postaci:

```
FontPath "/usr/X11R6/lib/X11/fonts/URW/"
```

Alternatywną metodą jest wpisanie w trakcie sesji X:

```
% xset fp+ /usr/X11R6/lib/X11/fonts/URW
% xset fp rehash
```

O ile rozwiązanie to również przyniesie pożądany efekt, o tyle dokonane w ten sposób zmiany zostaną stracone po zakończeniu sesji X. Oczywiście powyższe polecenia można dodać do pliku startowego (~/.xinitrc dla typowej sesji startx bądź pliku ~/.xsession przy logowaniu się za pomocą graficznego menedżera logowania, jak np. **XDM**). Trzecią metodą jest skorzystanie z nowego pliku /usr/X11R6/etc/fonts/local.conf: szczegóły w części poświęconej wyładowaniu.

5.5.2. Czcionki TrueType®

Xorg posiada wbudowaną obsługę czcionek TrueType. Istnieją dwa moduły, które mogą aktywować tę funkcję. W przykładzie wykorzystany został moduł freetype, z uwagi na większą spójność z innymi wewnętrznymi elementami wyświetlającymi czcionki. By włączyć moduł freetype, wystarczy dodać poniższy wiersz do sekcji "Module" pliku /etc/X11/xorg.conf.

```
Load "freetype"
```

Teraz musimy utworzyć katalog dla czcionek TrueType (na przykład /usr/X11R6/lib/X11/fonts/TrueType) i skopiować wszystkie czcionki do tego katalogu. Należy pamiętać, że czcionki TrueType nie mogą być bezpośrednio skopiowane z systemu Macintosh®, by możliwe było wykorzystanie ich z X11; muszą być w formacie UNIX/MS-DOS/Windows. Po skopiowaniu plików należy wykorzystać **ttmkfdir** do stworzenia pliku fonts.dir, by poinformować X, że zostały zainstalowane nowe czcionki. Program **ttmkfdir** dostępny jest z kolekcji portów FreeBSD jako x11-fonts/ttmkfdi.

```
# cd /usr/X11R6/lib/X11/fonts/TrueType
# ttmkfdi -o fonts.dir
```

Na koniec musimy dodać katalog TrueType do ścieżki czcionek. Robimy to analogicznie jak w przypadku czcionek Type1, za pomocą polecenia

```
% xset fp+ /usr/X11R6/lib/X11/fonts/TrueType
% xset fp rehash
```

bądź dodając wiersz FontPath do pliku xorg.conf.

Gotowe. Teraz **Netscape**, **Gimp**, **StarOffice™** i wszystkie inne aplikacje X powinny rozpoznawać zainstalowane czcionki TrueType. Bardzo małe czcionki (jak np. tekst na stronie WWW przy ustawionej wysokiej rozdzielczości ekranu) oraz bardzo duże (w **StarOffice**) będą wyglądały zdecydowanie lepiej.

5.5.3. Wygładzane czcionki

Zaktualizował Joe Marcus Clarke.

Wygładzanie (anti-aliasing) dostępne było w X11 od **XFree86** 4.0.2. Jednakże konfiguracja czcionek była niezmiernie nieporęczna do ukazania się **XFree86** 4.3.0. Począwszy od tej właśnie wersji, wszystkie czcionki w X11 dostępne w katalogach `/usr/X11R6/lib/X11/fonts/` oraz `~/.fonts/` są automatycznie dostępne dla aplikacji korzystających z wygładzania Xft. Nie wszystkie aplikacje potrafią korzystać z Xft, lecz wiele z czasem otrzymało wsparcie Xft. Przykładami aplikacji korzystających z Xft są Qt 2.3 i późniejsze (pakiet narzędzi graficznych dla środowiska **KDE**), GTK+ 2.0 i późniejsze (pakiet narzędzi graficznych dla środowiska **GNOME** desktop) oraz **Mozilla** 1.2 i późniejsze.

By móc kontrolować, które czcionki będą wygładzane bądź skonfigurować właściwości wygładzania, należy stworzyć (bądź zmodyfikować, jeśli istnieje) plik `/usr/X11R6/etc/fonts/local.conf`. Wykorzystując ten plik możemy dostosować kilka zaawansowanych opcji systemu czcionek Xft, jednakże rozdział ten skupia się jedynie na kilku podstawowych funkcjach. Więcej szczegółów znaleźć można w podręczniku systemowym `fonts-conf(5)`.

W pliku tym stosowany jest format kodu XML. Przy jego edycji należy pamiętać o właściwym zamykaniu wszystkich znaczników. Zaczyna się on od typowego nagłówka XML oraz definicji DOCTYPE. Następnym w kolejności jest znacznik `<fontconfig>`:

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
```

Jak już to zostało wcześniej powiedziane, wszystkie czcionki w katalogach `/usr/X11R6/lib/X11/fonts/` oraz `~/.fonts/` są automatycznie dostępne dla aplikacji korzystających z Xft. Jeśli natomiast chcemy dodać inny katalog nie będący podkatalogiem żadnego z powyższych, musimy dodać do pliku `/usr/X11R6/etc/fonts/local.conf` wiersz podobny do poniższego:

```
<dir>/ścieżka/do/moich/czcionek</dir>
```

Po dodaniu nowych czcionek, a szczególnie nowych katalogów, powinniśmy uruchomić poniższe polecenie, by przebudować bufor informacji o czcionkach:

```
# fc-cache -f
```

Wygładzanie sprawia, że brzegi czcionek stają się lekko zamazane, dzięki czemu małe litery są bardziej czytelne, a duże pozbawione efektu „schodków”. Może jednak prowadzić do zmęczenia oczu gdy zostanie użyte w stosunku do liter o normalnej wielkości. By wyłączyć wygładzanie czcionek o rozmiarze mniejszym niż 14 punktów, należy dodać poniższe wiersze do pliku konfiguracyjnego:

```
<match target="font">
  <test name="size" compare="less">
    <double>14</double>
  </test>
  <edit name="antialias" mode="assign">
```

```

        <bool>>false</bool>
    </edit>
</match>
<match target="font">
    <test name="pixelsize" compare="less" qual="any">
        <double>14</double>
    </test>
    <edit mode="assign" name="antialias">
        <bool>>false</bool>
    </edit>
</match>

```

Korzystając z wygładzania może się okazać, iż również odstępy pomiędzy literami niektórych czcionek o stałej szerokości są niewłaściwe. Ma to miejsce szczególnie w przypadku **KDE**. Jedynym rozwiązaniem tego problemu jest wymuszenie stałego odstępu o wartości 100 dla danych czcionek. W tym celu musimy wpisać:

```

<match target="pattern" name="family">
    <test qual="any" name="family">
        <string>fixed</string>
    </test>
    <edit name="family" mode="assign">
        <string>mono</string>
    </edit>
</match>
<match target="pattern" name="family">
    <test qual="any" name="family">
        <string>console</string>
    </test>
    <edit name="family" mode="assign">
        <string>mono</string>
    </edit>
</match>

```

(powyższe deklaruje inne typowe nazwy czcionek o stałej szerokości jako "mono"), a następnie:

```

<match target="pattern" name="family">
    <test qual="any" name="family">
        <string>mono</string>
    </test>
    <edit name="spacing" mode="assign">
        <int>100</int>
    </edit>
</match>

```

Niektóre czcionki, jak np. Helvetica, mogą stwarzać problemy jeśli zostaną poddane wygładzaniu. Z reguły daje to efekt czcionki przeciętej pionowo na pół. W najgorszym wypadku, może prowadzić to do załamania aplikacji typu **Mozilla**. By tego uniknąć, warto dodać poniższe do pliku `local.conf`:

```

<match target="pattern" name="family">
    <test qual="any" name="family">
        <string>Helvetica</string>
    </test>
    <edit name="family" mode="assign">

```

```

        <string>sans-serif</string>
    </edit>
</match>

```

Skończywszy edycję `local.conf` upewnijmy się, że plik kończy się znacznikiem `</fontconfig>`. Bez tego może się okazać, że nasze zmiany zostaną zignorowane.

Korzystanie z domyślnego zestawu czcionek, dostępnego wraz z X11, nie jest wskazane jeśli chodzi o wygładzanie. Zdecydowanie lepszy zestaw domyślnych czcionek zawiera port `x11-fonts/bitstream-vera`. Port ten zainstaluje również plik `/usr/X11R6/etc/fonts/local.conf` jeśli jeszcze go nie mamy. Jeśli natomiast istnieje już taki plik w systemie, stworzony zostanie plik `/usr/X11R6/etc/fonts/local.conf-vera`. Wystarczy dołączyć zawartość tego pliku do `/usr/X11R6/etc/fonts/local.conf`, by czcionki Bitstream automatycznie zastąpiły domyślne czcionki X11 Serif, Sans Serif i Monospaced.

Na koniec, użytkownicy mogą dodać swoją konfigurację poprzez własny plik `.fonts.conf`. W tym celu każdy użytkownik może stworzyć i zmodyfikować plik `~/.fonts.conf`. Również ten plik wykorzystuje format XML.

Ostatnia rzecz: osoby korzystające z monitorów LCD mogą pragnąć zastosować wygładzanie podpikselowe. W skrócie, w metodzie tej czerwone, zielone i niebieskie komponenty (oddzielone w płaszczyźnie poziomej) traktowane są oddzielnie, co poprawia rozdzielczość poziomą i przynosi radykalne efekty. By włączyć wygładzanie podpikselowe należy dodać poniższy wiersz do pliku `local.conf`:

```

<match target="font">
    <test qual="all" name="rgba">
        <const>unknown</const>
    </test>
    <edit name="rgba" mode="assign">
        <const>rgb</const>
    </edit>
</match>

```

Notatka: Zależnie od typu monitora, może się okazać, że będziemy musieli zastąpić `rgb` wartościami `bgr`, `vrgb` lub `vbgr`: poeksperymentujmy i sprawdźmy co da najlepszy efekt.

Wygładzanie powinno być aktywne przy następnym uruchomieniu serwera X. Tym nie mniej, programy muszą zostać poinformowane, by z niego korzystać. W chwili obecnej pakiet narzędzi Qt korzysta z wygładzania czcionek, tym samym również i całe środowisko **KDE**. **GTK+** oraz **GNOME** również można skonfigurować do pracy z wygładzanymi czcionkami poprzez aplet „Font” (Sekcja 5.7.1.3 zawiera szczegółowy opis). Domyślnie, **Mozilla** 1.2 i późniejsze będą automatycznie korzystać z wygładzania. By wyłączyć tę opcję, należy ponownie skompilować program z parametrem `-DWITHOUT_XFT`.

5.6. Menedżer pulpitów X

Napisał Seth Kingsley.

5.6.1. Omówienie

Menedżer pulpitów X (ang. X Display Manager **XDM**) jest opcjonalną częścią Systemu okien X, wykorzystywaną do zarządzania sesjami logowania. Znajduje on zastosowanie w kilku typach sytuacji, włączając w to zarówno minimalistyczne „Terminale X”, komputery prywatne, jak również ogromne sieciowe serwery graficzne. Z uwagi na fakt, iż System okien X jest niezależny od wykorzystywanej sieci jak i protokołu, istnieje wiele możliwych konfiguracji klientów i serwerów na różnych maszynach połączonych ze sobą za pomocą sieci. **XDM** dostarcza graficznego interfejsu pozwalającego wybrać, z którym serwerem się połączymy, jak i przeprowadzić autoryzację, np. za pomocą kombinacji loginu i hasła.

XDM można postrzegać jako narzędzie dostarczające użytkownikowi takich samych funkcjonalności jak `getty(8)` (szczegółowy opis zawiera Sekcja 22.3.2). Oznacza to, że to właśnie menedżer pulpitów w imieniu użytkownika dokonuje logowania do systemu i uruchamia menedżera sesji (z reguły menedżera okien). Następnie, **XDM** oczekuje aż program zakończy działanie, sygnalizując tym samym, że użytkownik skończył pracę i menedżer pulpitów powinien go wylogować z systemu. W tym momencie **XDM** może ponownie wyświetlić ekran logowania i wyboru środowiska graficznego oczekując na kolejnego użytkownika.

5.6.2. Korzystanie z XDM

Demon **XDM** znajduje się w `/usr/X11R6/bin/xdm`. Program ten może zostać uruchomiony w dowolnej chwili przez użytkownika `root` i od razu rozpocznie zarządzanie ekranami X w lokalnym systemie. Jeśli jednak **XDM** ma być uruchamiany przy każdym starcie systemu, najlepszym rozwiązaniem jest dodanie odpowiedniego wpisu do pliku `/etc/ttys`. Sekcja 22.3.2.1 zawiera więcej informacji odnośnie formatu i wykorzystania tego pliku. W domyślnej wersji plik `/etc/ttys` zawiera wiersz uruchamiający demona **XDM** w wirtualnym terminalu:

```
ttyv8      "/usr/X11R6/bin/xdm -nodaemon"  xterm    off secure
```

Domyślnie, wiersz ten jest nieaktywny. By go uaktywnić należy zmienić zawartość 5 kolumny z `off` na `on` i ponownie uruchomić `init(8)` wykorzystując wskazówki z Sekcja 22.3.2.2. Pierwsza kolumna - nazwa terminala, którym będzie zarządzał dany program - to `ttyv8`. Oznacza to, że **XDM** będzie pracował na dziewiątym wirtualnym terminalu.

5.6.3. Konfiguracja XDM

W katalogu `/usr/X11R6/lib/X11/xdm` znajdują się pliki konfiguracyjne **XDM**. Pliki te można wykorzystać do zmiany zachowania i wyglądu menedżera ekranów. Z reguły są to następujące pliki:

Plik	Opis
<code>Xaccess</code>	Zestaw reguł autoryzacji klientów.
<code>Xresources</code>	Domyślne wartości zasobów X.
<code>Xservers</code>	Lista zdalnych i lokalnych ekranów do zarządzania.
<code>Xsession</code>	Domyślny skrypt sesji logowania.

Plik	Opis
<code>Xsetup_*</code>	Skrypt uruchamiający aplikację przed interfejsem logowania.
<code>xdm-config</code>	Konfiguracja globalna dla wszystkich ekranów danej maszynie.
<code>xdm-errors</code>	Błędy generowane przez program serwera.
<code>xdm-pid</code>	Identyfikator procesu aktualnie działającego XDM.

W tym katalogu znajduje się również kilka skryptów i programów wykorzystywanych do konfiguracji pulpitów w trakcie działania **XDM**. Zadanie każdego z tych plików zostanie pokrótce omówione. Dokładna składnia i wykorzystanie wszystkich tych plików znajduje się w `xdm(1)`.

W domyślnej konfiguracji pojawi się prostokątne okno logowania z nazwą maszyny wypisaną dużą czcionką na samej górze, wraz z polami „Login:” i „Password:”. Jest to dobry punkt wyjściowy do modyfikacji wyglądu i zachowania ekranów **XDM**.

5.6.3.1. Xaccess

Protokół wykorzystywany do łączenia z pulpitemi obsługiwanymi przez **XDM** nosi nazwę X Display Manager Connection Protocol (XDMCP). Plik ten jest zestawem reguł do kontroli połączeń XDMCP ze zdalnych maszyn. Z reguły jest on ignorowany, chyba że w pliku `xdm-config` zostanie włączona opcja nasłuchiwania zdalnych połączeń. Domyślnie nie zezwala się na połączenia z innych klientów.

5.6.3.2. Xresources

Jest to plik domyślnej konfiguracji programu wyboru pulpitu i ekranu logowania. W tym właśnie pliku można modyfikować ich wygląd. Format pliku jest identyczny z formatem `app-defaults` opisanym w dokumentacji X11.

5.6.3.3. Xservers

Lista zdalnych pulpitów, do wyboru których mamy mieć dostęp za pomocą menedżera.

5.6.3.4. Xsession

Domyślny skrypt sesji **XDM** uruchamiany po zalogowaniu się użytkownika. Z reguły każdy użytkownik posiada zmodyfikowany według własnego upodobania skrypt sesji w pliku `~/.xsession`, uruchamiany zamiast tego skryptu.

5.6.3.5. Xsetup_*

Skrypty te zostaną automatycznie uruchomione przed wyświetleniem interfejsu logowania i wyboru pulpitu. Dla każdego wykorzystywanego ekranu znajduje się tu plik o nazwie `xsetup_` wraz z numerem lokalnego ekranu (na przykład `xsetup_0`). Z reguły skrypty te uruchamiają jeden bądź dwa programy w tle, jak np. `xconsole`.

5.6.3.6. xdm-config

Plik ustawień w formacie app-defaults, mający zastosowanie do wszystkich pulpitu zarządzanych przez menedżera.

5.6.3.7. xdm-errors

Plik ten zawiera wydruki wyjściowe serwerów X, które **XDM** stara się uruchomić. Jeśli w trakcie uruchamiania pulpitu z jakiegoś powodu proces ten zawiesi się, najlepszym miejscem poszukiwania komunikatów błędów jest właśnie ten plik. Komunikaty te są również umieszczane w pliku `~/.xsession-errors` użytkownika dla danej sesji.

5.6.4. Konfiguracja sieciowego serwera graficznego

By umożliwić innym klientom łączenie się z serwerem graficznym, należy zmodyfikować reguły kontroli dostępu i włączyć opcję nasłuchiwania połączeń. Domyślnie opcja ta jest nie aktywna. Jej aktywacja polega na odkomentowaniu poniższej linii w pliku `xdm-config`:

```
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:      0
```

Następnie należy ponownie uruchomić **XDM**. Pamiętajmy, że komentarze w plikach app-defaults rozpoczynają się od znaku „!” zamiast typowego „#”. Lektura przykładowego pliku `Xaccess` oraz podręcznika systemowego `xdm(1)` może nam pomóc gdy będziemy potrzebować bardziej surowej kontroli dostępu.

5.6.5. Alternatywy dla XDM

Dostępnych jest kilka alternatyw dla domyślnego menedżera **XDM**. Jeden z nich - **kdm** (dostarczany razem z **KDE**) - został bliżej opisany w tym w dalszej części rozdziału. Menedżer **kdm** oferuje wiele wizualnych usprawnień i kosmetycznych dodatków, jak również możliwość wyboru menedżera okien przed zalogowaniem do systemu.

5.7. Środowiska graficzne

Napisał Valentino Vaschetto.

Niniejsza sekcja opisuje różne typy środowisk graficznych dostępnych dla X we FreeBSD. Termin „środowisko graficzne” może oznaczać wszystko, od prostego menedżera okien po kompletny zestaw aplikacji pulpitu, jak **KDE** czy **GNOME**.

5.7.1. GNOME

5.7.1.1. O GNOME

GNOME jest przyjaznym użytkownikowi środowiskiem graficznym, umożliwiającym łatwą konfigurację i proste korzystanie z komputera. **GNOME** posiada panel (do uruchamiania aplikacji i wyświetlania ich statusu), pulpit

(gdzie można umieszczać dane i aplikacje), zestaw standardowych narzędzi biurowych i aplikacji oraz zestaw pewnych konwencji ułatwiających współpracę między aplikacjami i zachowanie wzajemnej spójności. Użytkownicy innych systemów operacyjnych powinni czuć się jak w domu korzystając z potężnego środowiska graficznego dostarczanego przez **GNOME**. Więcej informacji odnośnie środowiska **GNOME** we FreeBSD dostępnych jest na stronie WWW projektu FreeBSD GNOME Project (<http://www.FreeBSD.org/gnome>). Strona ta zawiera również w miarę zrozumiałe FAQ traktujące o instalacji, konfiguracji i zarządzaniu **GNOME**.

5.7.1.2. Instalacja GNOME

Najprostszym sposobem instalacji **GNOME** jest poprzez menu „Desktop Configuration” w trakcie instalacji FreeBSD, co omawia Sekcja 2.9.13 rozdziału 2. Możliwa jest również instalacja z pakietu bądź kolekcji portów:

By zainstalować pakiet **GNOME** z sieci, wystarczy wpisać:

```
# pkg_add -r gnome2
```

By skompilować **GNOME** ze źródeł najlepiej jest skorzystać z portu:

```
# cd /usr/ports/x11/gnome2
# make install clean
```

Mając już zainstalowanego **GNOME** musimy poinformować serwer X, by uruchamiał właśnie jego w miejsce domyślnego menedżera okien.

Najprostszą metodą uruchomienia **GNOME** jest wykorzystanie **GDM** - menedżera pulpitów GNOME (ang. GNOME Display Manager). **GDM** jest instalowany jako część środowiska **GNOME**, jednakże jest on domyślnie wyłączony. By go włączyć, należy dodać wiersz `gdm_enable="YES"` do pliku `/etc/rc.conf`. Po ponownym uruchomieniu systemu, **GNOME** zostanie automatycznie włączony zaraz po zalogowaniu się — nie wymagana jest dodatkowa konfiguracja.

Oczywiście, **GNOME** można uruchomić również bezpośrednio z linii poleceń poprawnie konfigurując plik `.xinitrc` w katalogu domowym. Jeśli plik ten już istnieje wystarczy zastąpić wiersz odpowiadający za uruchomienie aktualnego menedżera okien na wiersz uruchamiający `/usr/X11R6/bin/gnome-session`. Jeśli w pliku nie dokonywaliśmy żadnych istotnych zmian, najprościej będzie po prostu wpisać:

```
% echo "/usr/X11R6/bin/gnome-session" > ~/.xinitrc
```

Następnie wpisujemy `startx`, co spowoduje uruchomienie środowiska **GNOME**.

Notatka: Jeśli wykorzystujemy starszego menedżera okien, jak np. **XDM**, powyższe rozwiązanie nie zadziała. W takiej sytuacji musimy stworzyć plik wykonywalny `.xsession` zawierający powyższe polecenie. W tym celu należy zmodyfikować ten plik i zastąpić polecenie uruchamiające aktualnego menedżera poleceniem `/usr/X11R6/bin/gnome-session`:

```
% echo "#!/bin/sh" > ~/.xsession
% echo "/usr/X11R6/bin/gnome-session" >> ~/.xsession
% chmod +x ~/.xsession
```

Jeszcze jedną metodą jest skonfigurowanie menedżera pulpitów tak, by umożliwiał wybór menedżera okien w trakcie logowania. Sekcja Więcej informacji o KDE wyjaśnia jak to zrobić w **kdm** - menedżerze pulpitów **KDE**.

5.7.1.3. Wygładzane czcionki w GNOME

X11 obsługuje wygładzanie czcionek (anti-aliasing) za pomocą rozszerzenia „RENDER”. GTK+ w wersji 2.0 i późniejszych (pakiet narzędzi graficznych wykorzystywany przez **GNOME**) potrafi korzystać z tej funkcji. Konfigurację wygładzania czcionek opisuje Sekcja 5.5.3. Zatem, wykorzystując najnowsze oprogramowanie, możliwe jest wygładzanie czcionek w środowisku **GNOME**. Wystarczy przejść do menu Applications→Desktop Preferences→Font i wybrać jedną z opcji: **Best shapes** (najlepsze kształty), **Best contrast** (najlepszy kontrast) lub **Subpixel smoothing (LCDs)** (wygładzanie podpikselowe). Natomiast dla aplikacji GTK+ nie będących częścią środowiska **GNOME**, należy ustawić zmienną środowiskową `GDK_USE_XFT` na 1 przed uruchomieniem programu.

5.7.2. KDE

5.7.2.1. O KDE

KDE jest prostym w użyciu współczesnym środowiskiem graficznym, zawierającym między innymi:

- Ładnie wyglądający pulpit
- Pulpit odznaczający się całkowitą przezroczystością sieci
- Zintegrowany system pomocy, udostępniający w prosty sposób informacje o korzystaniu ze środowiska **KDE** i jego aplikacji
- Jednakowy wygląd i zachowanie wszystkich aplikacji **KDE**
- Standardowe menu i paski narzędzi, skróty klawiaturowe, schematy kolorów, itp.
- Internacjonalizacja: **KDE** jest dostępny w ponad 40 językach
- Scentralizowaną i spójną konfigurację środowiska
- Całą masę przydanych aplikacji

KDE posiada własną przeglądarkę internetową - **Konqueror**, która stanowi poważną konkurencję dla innych przeglądarek z systemów UNIX. Więcej informacji o **KDE** znaleźć można na stronie KDE (<http://www.kde.org/>). Natomiast informacje o jego współpracy z FreeBSD dostępne są na stronie FreeBSD-KDE team (<http://freebsd.kde.org/>).

5.7.2.2. Instalacja KDE

Podobnie jak w przypadku **GNOME** czy dowolnego innego środowiska graficznego, najprostszym sposobem instalacji **KDE** jest skorzystanie z menu „Desktop Configuration” w procesie instalacji FreeBSD, co omawia Sekcja 2.9.13 rozdziału 2. Ponownie, również **KDE** można zainstalować z pakietu bądź z kolekcji portów:

By zainstalować pakiet **KDE** z sieci, wystarczy wpisać:

```
# pkg_add -r kde
```

`pkg_add(1)` automatycznie pobierze najnowszą wersję aplikacji.

By skompilować **KDE** ze źródeł najlepiej jest skorzystać z portu:

```
# cd /usr/ports/x11/kde3
```

```
# make install clean
```

Po instalacji **KDE** należy poinformować serwer X, by uruchamiał go w miejsce domyślnego menedżera okien. W tym celu należy zmodyfikować plik `.xinitrc`:

```
% echo "exec startkde" > ~/.xinitrc
```

Od tej pory, za każdym razem gdy uruchomimy System okien X za pomocą polecenia `startx`, uruchomione zostanie środowisko **KDE**.

Jeśli wykorzystujemy starszego menedżera okien, jak np. **XDM**, wymagana jest odmienna konfiguracja. Opis konfiguracji **kdm** znajduje się w dalszej części tego rozdziału.

5.7.3. Więcej informacji o KDE

Skoro zainstalowaliśmy już **KDE**, większość informacji można odnaleźć w systemie pomocy, bądź po prostu klikając w dowolne menu. Użytkownicy systemów Windows czy Mac® powinni czuć się jak w domu.

Najlepszym źródłem informacji o **KDE** jest dostępna w sieci dokumentacja. **KDE** zawiera własną przeglądarkę internetową - **Konqueror**, masę przydatnych aplikacji i obszerną dokumentację. Pozostała część tego rozdziału skupi się na technicznych zagadnieniach, trudnych do nauczenia się poprzez dość losowe poznawanie środowiska.

5.7.3.1. Menedżer pulpitu KDE

Administratorzy systemów wieloużytkownikowych mogą chcieć skorzystać z graficznego ekranu logowania. W tym celu można zastosować **XDM**, jak to zostało opisane wcześniej. Można również wykorzystać alternatywne rozwiązanie dostępne razem z **KDE** - **kdm** - wyglądające zdecydowanie bardziej atrakcyjnie oraz posiadające wiele dodatkowych opcji logowania. W szczególności, użytkownicy mogą w prosty sposób wybrać (poprzez menu), które środowisko graficzne uruchomić po zalogowaniu (**KDE**, **GNOME**, bądź inne).

By aktywować **kdm**, należy zmodyfikować wpis dla `tttyv8` w pliku `/etc/ttys`. Wiersz ten powinien wyglądać następująco:

```
tttyv8 "/usr/local/bin/kdm -nodaemon" xterm on secure
```

5.7.4. XFce

5.7.4.1. O XFce

XFce jest środowiskiem graficznym wykorzystującym pakiet narzędzi **GTK+**, podobnie jak **GNOME**, lecz jest zdecydowanie lżejsze i przeznaczone dla osób poszukujących prostego i efektywnego środowiska, lecz również łatwego w obsłudze i konfiguracji. Wyglądem bardzo przypomina **CDE**, często dostępne w komercyjnych systemach UNIX. Niektóre z cech **XFce**:

- Prosty i łatwy w obsłudze pulpit
- Możliwość konfiguracji wszystkich elementów za pomocą myszki, metody przeciągnij i upuść, itp.
- Główny panel podobny do **CDE** z wieloma opcjami menu, apletami i programami wywołującymi

- Zintegrowane menedżery okien, plików, dźwięku, moduł zgodności **GNOME** i inne dodatki
- Możliwość stosowania motywów (skoro wykorzystuje GTK+)
- Szybkie, lekkie i wydajne: idealny dla starszych/wolniejszych maszyn lub z ograniczonym zasobem pamięci

Więcej informacji dostępnych jest na stronie XFce (<http://www.xfce.org/>).

5.7.4.2. Instalacja XFce

W chwili pisania niniejszego tekstu dostępny jest pakiet binarny. By z niego zainstalować **XFce**, wystarczy wpisać:

```
# pkg_add -r xfce4
```

Oczywiście, można również skompilować go ze źródeł przy pomocy kolekcji portów:

```
# cd /usr/ports/x11-wm/xfce4
# make install clean
```

Pozostaje jeszcze poinformować serwer X by uruchamiał **XFce** przy kolejnych uruchomieniach X. Wystarczy wpisać:

```
% echo "/usr/X11R6/bin/startxfce4" > ~/.xinitrc
```

Przy kolejnym uruchomieniu X jako środowisko graficzne zostanie wykorzystane **XFce**. Podobnie jak wcześniej, tak i teraz należy stworzyć plik `.xsession` gdy korzystamy z **XDM**, co zostało umówione w części poświęconej **GNOME**, wpisując polecenie `/usr/X11R6/bin/startxfce4`. Alternatywnie, należy skonfigurować menedżera pulpitu, by pozwalał na wybór środowiska graficznego w trakcie logowania, zgodnie z opisem z sekcji poświęconej `kdm`.

II. Codzienne czynności

Skoro podstawy zostały już omówione, ta część Podręcznika zajmie się kilkoma z najczęściej wykorzystywanych funkcji FreeBSD. Niniejsze rozdziały:

- Przedstawią popularne i przydatne aplikacje biurowe: przeglądarki, edytory dokumentów, itp.
- Przedstawią narzędzia multimedialne dostępne dla FreeBSD.
- Wyjaśnią proces kompilacji własnego jądra FreeBSD w celu włączenia dodatkowych funkcji w systemie.
- Opiszą szczegółowo system wydruku, zarówno dla drukarek podłączonych lokalnie jak i drukarek sieciowych.
- Pokażą jak uruchomić aplikacje Linuksowe w systemie FreeBSD.

Niektóre z poniższych rozdziałów zalecają lekturę dodatkowych materiałów, co zostanie wskazane w streszczeniu na początku każdego rozdziału.

Rozdział 6. Aplikacje biurowe

Napisał Christophe Juniet. Tłumaczył Cezary Morga.

6.1. Streszczenie

Podobnie jak we wszystkich współczesnych systemach operacyjnych, również i we FreeBSD możemy uruchamiać szereg aplikacji biurowych, jak np. przeglądarki czy procesory tekstu. Większość z nich dostępnych jest zarówno w postaci pakietów jak i portów. Rozdział ten zaprezentuje jak bez większego wysiłku można je zainstalować zarówno z odpowiednich pakietów jak też wprost z kolekcji portów.

Pamiętajmy, że instalacja programów z portów obejmuje również ich kompilację ze źródeł. Stąd też proces ten może zająć dużo czasu, zależnie od tego co kompilujemy, oraz od mocy obliczeniowej naszej maszyny. Jeśli kompilacja ze źródeł jest dla nas zbyt czasochłonnym zadaniem, większość programów dostępnych w kolekcji portów możemy zainstalować również z prekompilowanych pakietów.

Jako, że FreeBSD umożliwia tzw. tryb zgodności binarnej z Linuksem, wiele aplikacji pisanych pod Linuksa dostępnych jest również we FreeBSD. Jednakże, przed instalacją jakiegokolwiek programu linuxowego zalecamy przeczytać Rozdział 10 niniejszego Podręcznika. Nazwy wielu portów wykorzystujących zgodność binarną z Linuksem rozpoczynają się od „linux-”, o czym warto pamiętać poszukując właściwego portu, np. za pomocą polecenia `whereis(1)`. W dalszej części rozdziału założono, że przed instalacją jakiegokolwiek linuxowej aplikacji w naszym komputerze został włączony tryb zgodności z Linuksem.

Programy omówione w tym rozdziale zostały podzielone na następujące kategorie:

- Przeglądarki internetowe (takie jak **Mozilla**, **Opera**, **Firefox** czy **Konqueror**)
- Programy codziennego użytku (jak np. **KOffice**, **AbiWord**, **The GIMP** oraz **OpenOffice.org**)
- Przeglądarki dokumentów (takie jak **Acrobat Reader®**, **gv**, **Xpdf** i **GQview**)
- Finanse (jak np. **GnuCash**, **Gnumeric**, **Abacus**)

Przed przeczytaniem tego rozdziału, powinniśmy:

- Wiedzieć jak instalować dodatkowe programy (Rozdział 4).
- Wiedzieć, jak instalować programy linuxowe (Rozdział 10).

Rozdział 7 zawiera informacje odnośnie instalacji środowiska multimedialnego. Natomiast Rozdział 24 zawiera wskazówki jak skonfigurować i korzystać z poczty elektronicznej.

6.2. Przeglądarki internetowe

FreeBSD z definicji nie posiada zainstalowanej żadnej przeglądarki internetowej. W zamian katalog `www` (<http://www.FreeBSD.org/ports/www.html>) kolekcji portów zawiera całą masę przeglądarek gotowych do instalacji. Jeśli nie mamy czasu na kompilację (co w niektórych przypadkach może zająć naprawdę dużo czasu), wiele z nich udostępnionych zostało również w postaci pakietów.

KDE i **GNOME** dysponują własnymi przeglądarkami internetowymi. Sekcja 5.7 zawiera szczegółowe informacje odnośnie instalacji tych środowisk graficznych.

Jeśli szukamy lekkich przeglądarek internetowych, powinniśmy zainteresować się `www/dillo`, `www/links` lub `www/w3m`.

Niniejsza sekcja omawia następujące programy:

Nazwa aplikacji	Wykorzystanie zasobów	Instalacja z portów	Główne zależności
Mozilla	duże	długa	Gtk+
Opera	małe	krótka	Dostępne są wersje dla FreeBSD i Linuksa. Wersja dla Linuksa wymaga trybu zgodności binarnej z Linuksem oraz linux-openmotif .
Firefox	średnie	długa	Gtk+
Konqueror	średnie	długa	Biblioteki KDE

6.2.1. Mozilla

Mozilla jest nowoczesną, stabilną przeglądarką w całości przeniesioną na FreeBSD. Zawiera w pełni zgodny ze standardami mechanizm wyświetlania kodu HTML, jak również klienta poczty elektronicznej i grup dyskusyjnych. Dysponuje nawet edytorem HTML, jeśli sami chcemy pisać strony internetowe. Użytkownicy **Netscape** z pewnością dostrzegą podobieństwo do pakietu **Communicator**, gdyż obydwie przeglądarki mają te same pochodzenie.

Na wolnych maszynach, z procesorem wolniejszym niż 233MHz bądź z pojemnością pamięci RAM mniejszą niż 64MB, **Mozilla** może okazać się zbyt „zasobo-żerna”. W tej sytuacji możemy zainteresować się np. przeglądarką **Opera**, opisaną w dalszej części tego rozdziału.

Jeśli nie możemy bądź z dowolnego powodu nie chcemy kompilować przeglądarki **Mozilla**, grupa FreeBSD GNOME zrobiła to za nas. Wystarczy zainstalować pakiet bezpośrednio z sieci za pomocą:

```
# pkg_add -r mozilla
```

Jeśli z jakichś powodów pakiet nie jest dostępny, a my dysponujemy czasem i miejscem na dysku, możemy pobrać źródła, skompilować je i zainstalować w naszym systemie. W tym celu wystarczy wpisać:

```
# cd /usr/ports/www/mozilla
# make install clean
```

Port ten przygotowany został w sposób zapewniający właściwą inicjalizację poprzez uruchamianie rejestru konfiguracji z uprawnieniami użytkownika `root`, w momencie gdy pracujemy na koncie zwykłego użytkownika. Tym nie mniej, jeśli chcemy poprawnie zainstalować dodatkowe składniki, musimy uruchomić program **Mozilla** jako `root`.

By uruchomić przeglądarkę należy wpisać poniższe polecenie. Poza procesem instalacji, przeglądarka nie wymaga korzystania z konta `root`.

```
% mozilla
```

Uruchomienie jej bezpośrednio w trybie klienta poczty i grup dyskusyjnych możliwe jest za pomocą polecenia:

```
% mozilla -mail
```

6.2.2. Firefox

Firefox jest nowoczesną przeglądarką, opartą o kod przeglądarki **Mozilla**. O ile **Mozilla** stanowi kompletny pakiet aplikacji - zawiera m.in. przeglądarkę, klienta poczty, czy grup dyskusyjnych, o tyle **Firefox** jest jedynie przeglądarką, dzięki czemu jest zdecydowanie mniejszy i szybszy.

By zainstalować go z pakietu wystarczy wpisać:

```
# pkg_add -r firefox
```

Jeśli preferujemy kompilację programów wprost z kodu źródłowego, możemy skorzystać z kolekcji portów:

```
# cd /usr/ports/www/firefox
# make install clean
```

6.2.3. Firefox, Mozilla i moduł Java™

Notatka: W tej i następnej sekcji założono, że mamy już zainstalowaną przeglądarkę **Firefox** lub **Mozilla**.

Fundacja FreeBSD posiada licencję Sun Microsystems na dystrybucję plików binarnych FreeBSD dla środowisk Java Runtime Environment (JRE™) oraz Java Development Kit (JDK™). Pakiety binarne dla FreeBSD dostępne są na stronie WWW Fundacji FreeBSD (<http://www.freebsdoundation.org/downloads/java.shtml>).

By do przeglądarki **Firefox** lub **Mozilla** dodać obsługę Java™, musimy wpierw zainstalować port `java/javavmwrapper`, a następnie pobrać pakiet **Diablo JRE** ze strony <http://www.freebsdoundation.org/downloads/java.shtml>, i zainstalować go za pomocą `pkg_add(1)`.

Po ponownym uruchomieniu przeglądarki, wpisaniu w pasku adresu `about:plugins` i wciśnięciu **Enter**, wyświetlona zostanie strona informująca o zainstalowanych modułach. Wymieniony powinien zostać również moduł **Java**.

6.2.4. Firefox, Mozilla i moduł Macromedia® Flash™

Moduł Macromedia® Flash™ niestety nie jest dostępny dla FreeBSD. Tym nie mniej, istnieje interfejs programowy (ang. wrapper) do uruchamiania linuxowej wersji modułu. Interfejs ten obsługuje również moduły Adobe® Acrobat®, RealPlayer i wiele innych.

By zainstalować port `www/linuxpluginwrapper`, musimy wpierw zainstalować `emulators/linux_base`, który jest obszernym portem. W trakcie instalacji należy zwrócić szczególną uwagę na informacje o właściwej konfiguracji pliku `/etc/libmap.conf`! Przykładowe pliki konfiguracyjne znaleźć można w katalogu `/usr/local/share/examples/linuxpluginwrapper/`.

Kolejnym krokiem jest instalacja portu `www/linux-flashplugin7`. Po zainstalowaniu modułu możemy sprawdzić listę aktualnie dostępnych modułów uruchamiając przeglądarkę, wpisując w pasku adresu `about:plugins` i wciskając **Enter**.

Jeśli na powyższej liście brak jest modułu Flash, najczęstszą przyczyną jest brak odpowiedniego dowiązania symbolicznego. W takiej sytuacji należy jako użytkownik `root` uruchomić następujące polecenia:

```
# ln -s /usr/local/lib/npapi/linux-flashplugin/libflashplayer.so \
```

```

/usr/X11R6/lib/browser_plugins/
# ln -s /usr/local/lib/npapi/linux-flashplugin/flashplayer.xpt \
/usr/X11R6/lib/browser_plugins/

```

Po ponownym uruchomieniu przeglądarki, moduł powinien zostać wyświetlony na wspomnianej liście. Może się również zdarzyć, że nasza przeglądarka ulegnie awarii w trakcie odtwarzania animacji Flash. W takim przypadku będziemy musieli nałożyć odpowiednią łatę (ang. patch):

```

# cd /usr/src
# fetch http://people.FreeBSD.org/~nork/rtld_dlsym_hack.diff
# patch < rtld_dlsym_hack.diff
# cd libexec/rtld-elf/
# make clean
# make obj
# make depend
# make && make install

```

Po czym musimy ponownie uruchomić komputer.

Notatka: Port `linuxpluginwrapper` działa poprawnie jedynie na maszynach o architekturze i386.

6.2.5. Opera

Opera jest nowoczesną, zgodną ze standardami przeglądarką internetową. Posiada również klienta poczty elektronicznej i grup dyskusyjnych, klienta sieci IRC, czytnik wiadomości RSS/Atom i wiele innych. Mimo to **Opera** jest stosunkowo lekką i bardzo szybką przeglądarką. Dostępne są dwie wersje: wersja przeznaczona dla FreeBSD oraz wersja uruchamiana w trybie emulacji Linuksa.

By móc przeglądać zasoby sieci WWW za pomocą wersji dla FreeBSD, musimy zainstalować odpowiedni pakiet:

```
# pkg_add -r opera
```

Niektóre serwery FTP nie zawierają wszystkich pakietów, lecz ten sam efekt możemy otrzymać wykorzystując kolekcję portów:

```

# cd /usr/ports/www/opera
# make install clean

```

By zainstalować wersję linuxową należy w powyższych przykładach zmienić nazwę `opera` na `linux-opera`. Wersja linuxowa przydatna jest w sytuacjach wymagających modułów dostępnych tylko dla Linuksa, jak np. **Adobe Acrobat Reader**. Pod każdym innym względem wersje dla FreeBSD i Linuksa zdają się być funkcjonalnie identyczne.

6.2.6. Konqueror

Konqueror jest częścią środowiska graficznego **KDE**, lecz może być również wykorzystywane poza nim poprzez zainstalowanie `x11/kdebase3`. **Konqueror** jest więcej niż przeglądarką internetową, jest również menedżerem plików i przeglądarką plików multimedialnych.

Konqueror dostępny jest również z pakietem modułów, z portu `misc/konq-plugins`.

Również **Konqueror** obsługuje technologię **Flash**. Dokument opisujący instalację modułu dostępny jest pod adresem <http://freebsd.kde.org/howto.php>.

6.3. Programy codziennego użytku

Jeśli chodzi o programy codziennego użytku, pierwszą rzeczą, której często poszukuje wielu nowych użytkowników, jest dobry pakiet biurowy bądź po prostu procesor tekstu. Pomimo, że niektóre środowiska graficzne, jak np. **KDE**, dysponują własnym pakietem biurowym, nie istnieje żadna domyślna aplikacja. Niezależnie od wykorzystywanego środowiska graficznego, FreeBSD dysponuje wszystkim czego możemy potrzebować.

Sekcja ta omawia następujące aplikacje:

Nazwa aplikacji	Wykorzystanie zasobów	Instalacja z portów	Główne zależności
KOffice	małe	długa	KDE
AbiWord	małe	krótka	Gtk+ bądź GNOME
The Gimp	małe	długa	Gtk+
OpenOffice.org	duże	długa	JDK 1.4, Mozilla

6.3.1. KOffice

Spółeczność KDE udostępnia swoje środowisko graficzne wraz z pakietem biurowym, z którego można korzystać zarówno w **KDE** jak i poza nim. Zawiera cztery standardowe komponenty, które można odnaleźć również w innych pakietach biurowych: procesor tekstu **KWord**, arkusz kalkulacyjny **KSpread**, menedżer prezentacji multimedialnych **KPresenter** oraz program do tworzenia graficznych dokumentów - **Kontour**.

Przed instalacją najnowszej wersji pakietu **KOffice**, powinniśmy się upewnić, że dysponujemy również najnowszą wersją **KDE**.

By zainstalować **KOffice** z pakietu, należy wpisać następujące polecenie:

```
# pkg_add -r koffice
```

Jeśli pakiet nie jest dostępny, możemy wykorzystać kolekcję portów. Na przykład, by zainstalować **KOffice** dla **KDE3**, należy wpisać:

```
# cd /usr/ports/editors/koffice-kde3
# make install clean
```

6.3.2. AbiWord

AbiWord jest darmowym procesorem tekstu pod względem wyglądu i obsługi podobnym do **Microsoft Word**. Za jego pomocą możemy pisać artykuły, listy, raporty, notatki itp. Jest on bardzo szybki, bogaty w różnorodne funkcje i przyjazny użytkownikowi.

AbiWord potrafi importować z i eksportować do wielu formatów plików, w tym również niektórych własnościowych formatów, jak np. **Microsoft .doc**.

AbiWord dostępny jest w postaci pakietu. By go zainstalować wystarczy wpisać:

```
# pkg_add -r abiword
```

Jeśli pakiet nie jest dostępny, możemy skompilować program wprost z kolekcji portów:

```
# cd /usr/ports/editors/abiword
# make install clean
```

6.3.3. The GIMP

The GIMP jest wyrafinowanym programem przetwarzającym obraz. Wykorzystywany może być zarówno jako prosty program malujący jak i zaawansowny pakiet do retuszu fotografii. Obsługuje on dużą liczbę dodatkowych modułów, jak również udostępnia odpowiedni interfejs dla skryptów. **The GIMP** potrafi odczytywać i zapisywać wiele formatów plików. Obsługuje również interfejsy skanerów i tabletów.

Możemy zainstalować go z pakietu, za pomocą polecenia:

```
# pkg_add -r gimp
```

Jeśli wykorzystywany serwer FTP nie dysponuje odpowiednim pakietem, możemy wykorzystać kolekcję portów. Katalog graphics (<http://www.FreeBSD.org/ports/graphics.html>) zawiera oprócz samego programu, również podręcznik **The Gimp Manual**. Oto przykładowa metoda instalacji:

```
# cd /usr/ports/graphics/gimp
# make install clean
# cd /usr/ports/graphics/gimp-manual-pdf
# make install clean
```

Notatka: Wspomniany katalog graphics (<http://www.FreeBSD.org/ports/graphics.html>) kolekcji portów zawiera również wersję rozwojową aplikacji **The GIMP** pod nazwą `graphics/gimp-devel`. Wersja HTML podręcznika **The Gimp Manual** dostępna jest z portu `graphics/gimp-manual-html`.

6.3.4. OpenOffice.org

OpenOffice.org zawiera wszystkie aplikacje, które powinny znaleźć się w kompletnym pakiecie biurowym: procesor tekstu, arkusz kalkulacyjny, menedżer prezentacji i program do rysowania. Jego interfejs jest zbliżony do interfejsów innych pakietów biurowych. Może on importować i eksportować wiele popularnych formatów plików. Dostępny jest w wielu wersjach językowych interfejsu, narzędzi sprawdzania pisowni i słowników.

Procesor tekstu pakietu **OpenOffice.org** wykorzystuje format pliku XML, by tym sposobem zwiększyć przenośność i elastyczność dokumentów. Arkusz kalkulacyjny oferuje język makr, jak również obsługę interfejsów do zewnętrznych baz danych. **OpenOffice.org** jest stabilną aplikacją, dostępną dla platform Windows, Solaris™, Linux, FreeBSD, i Mac OS X. Więcej informacji o pakiecie **OpenOffice.org** znaleźć można na stronie OpenOffice.org (<http://www.openoffice.org/>). Informacje odnośnie wersji dla FreeBSD oraz możliwości bezpośredniego pobrania pakietów dostępne są na stronie WWW FreeBSD OpenOffice.org Porting Team (<http://porting.openoffice.org/freebsd/>).

By zainstalować **OpenOffice.org**, wystarczy:

```
# pkg_add -r openoffice.org
```

Notatka: Metoda ta przewidziana jest dla wydań FreeBSD gałęzi -RELEASE. W innym przypadku możemy być zmuszeni odwiedzić wspomnianą wyżej stronę WWW FreeBSD **OpenOffice.org** Porting Team, by pobrać a następnie zainstalować właściwy pakiet za pomocą `pkg_add(1)`. Dostępna jest zarówno wersja bieżąca jak i rozwojowa.

Mając zainstalowane pakiety, wystarczy wpisać następujące polecenie by uruchomić **OpenOffice.org**:

```
% openoffice.org
```

Notatka: Przy pierwszym uruchomieniu będziemy poproszeni o udzielenie kilku odpowiedzi. po czym w naszym katalogu macierzystym zostanie utworzony katalog `.openoffice.org2`.

Jeśli pakiety **OpenOffice.org** nie są dostępne, wciąż mamy możliwość skompilowania portu. Miejmy jednakże w pamięci, że wymaga to dużej ilości wolnej przestrzeni na dysku oraz zajmuje dość dużo czasu.

```
# cd /usr/ports/editors/openoffice.org-2
# make install clean
```

Notatka: Jeśli chcemy skompilować pakiet w naszej wersji językowej, należy powyższe polecenie zastąpić następującym:

```
# make LOCALIZED_LANG=nasz_jezyk install clean
```

Opcję `nasz_jezyk` należy zastąpić właściwym kodem ISO. Lista kodów obsługiwanych języków dostępna jest w pliku `files/Makefile.localized`, znajdującym się w katalogu portu.

Skończywszy instalację, możemy uruchomić **OpenOffice.org** za pomocą polecenia:

```
% openoffice.org
```

6.4. Przeglądarki dokumentów

Ostatnio na popularności zyskały niektóre z pośród nowych formatów dokumentów, przy czym niezbędne przeglądarki mogą nie być dostępne w podstawowej konfiguracji systemu. W tej sekcji opiszemy jak je zainstalować.

Niniejsza sekcja omawia następujące programy:

Nazwa aplikacji	Wykorzystanie zasobów	Instalacja z portów	Główne zależności
Acrobat Reader	małe	krótka	Tryb zgodności binarnej z Linuksem

Nazwa aplikacji	Wykorzystanie zasobów	Instalacja z portów	Główne zależności
gv	małe	krótka	Xaw3d
Xpdf	małe	krótka	FreeType
GQview	małe	krótka	Gtk+ lub GNOME

6.4.1. Acrobat Reader®

Obecnie wiele dokumentów publikowanych jest w postaci plików PDF (ang. Portable Document Format). Jedną z zalecanych przeglądarek do tego typu dokumentów jest **Acrobat Reader**, wydany przez firmę Adobe na platformę linuxową. We FreeBSD możemy uruchomić ją dzięki trybowi zgodności binarnej z Linuksem.

By zainstalować **Acrobat Reader 7** wprost z kolekcji portów, należy wpisać:

```
# cd /usr/ports/print/acroread7
# make install clean
```

Z uwagi na ograniczenia licencyjne, **Acrobat Reader** nie jest dostępny w postaci pakietu.

6.4.2. gv

gv jest przeglądarką dokumentów PostScript i PDF. Bazuje ona bezpośrednio na **ghostview**, lecz dzięki bibliotece **Xaw3d** wygląda zdecydowanie lepiej. **gv** jest szybką przeglądarką o przejrzystym interfejsie. Posiada wiele funkcji, jak np. możliwość ustawienia orientacji tekstu, rozmiaru papieru, skali czy wygładzania czcionek. Prawie każdą czynność można wykonać za pomocą klawiatury bądź myszki.

By zainstalować **gv** z pakietu, wystarczy wpisać:

```
# pkg_add -r gv
```

Jeśli nie możemy pobrać pakietu, możemy zawsze wykorzystać kolekcję portów:

```
# cd /usr/ports/print/gv
# make install clean
```

6.4.3. Xpdf

Jeśli potrzebujemy małej przeglądarki dokumentów PDF, **Xpdf** stanowi lekkie i wydajne rozwiązanie. Wymaga ona małej ilości zasobów i jest bardzo stabilna. Do pracy wykorzystuje standardowe czcionki X i nie wymaga **Motif**, ani żadnego innego pakietu narzędzi X.

By zainstalować pakiet **Xpdf**, należy wykorzystać następujące polecenie:

```
# pkg_add -r xpdf
```

Jeśli pakiet nie jest dostępny bądź wolimy wykorzystać kolekcję portów, wystarczy wpisać:

```
# cd /usr/ports/graphics/xpdf
# make install clean
```

Zakończywszy instalację, możemy uruchomić **Xpdf**. Menu dostępne jest za pomocą prawego przycisku myszki.

6.4.4. GQview

GQview jest menedżerem i przeglądarką obrazów. Za pomocą jednego kliknięcia możemy przeglądać pliki graficzne, uruchomić zewnętrzny edytor, uzyskać podgląd miniatur i wiele więcej. Mamy również dostęp do trybu pokazu slajdów oraz kilku podstawowych operacji na plikach. Możemy łatwo zarządzać kolekcjami obrazów i odnajdywać powtarzające się pliki. **GQview** udostępnia również tryb pełnoekranowy oraz obsługę wielu języków.

By zainstalować pakiet **GQview**, wystarczy wpisać:

```
# pkg_add -r gqview
```

Jeśli pakiet nie jest dostępny bądź wolimy skorzystać z kolekcji portów, możemy wpisać:

```
# cd /usr/ports/graphics/gqview
# make install clean
```

6.5. Finanse

Jeśli z jakiegoś powodu chcielibyśmy zarządzać naszym domowym budżetem we FreeBSD, dostępnych mamy kilka rozbudowanych i łatwych w obsłudze aplikacji. Niektóre z nich są zgodne z szeroko rozpowszechnionymi formatami plików jak np. dokumenty **Quicken®** czy **Excel**.

Sekcja ta omawia następujące aplikacje:

Nazwa aplikacji	Wykorzystanie zasobów	Instalacja z portów	Główne zależności
GnuCash	małe	długa	GNOME
Gnumeric	małe	długa	GNOME
Abacus	małe	krótka	Tcl/Tk

6.5.1. GnuCash

GnuCash jest efektem usilnych starań środowiska **GNOME** by dostarczać końcowym użytkownikom przyjazne i rozbudowane aplikacje. Za pomocą **GnuCash** możemy śledzić nasze przychody i wydatki, stan konta bankowego czy papierów wartościowych. Posiada on intuicyjny interfejs pozostając wciąż zaawansowanym narzędziem.

GnuCash zawiera inteligentny rejestr, hierarchiczny system kont, wiele skrótów klawiaturowych i metody autouzupełniania wprowadzanych danych. Umożliwia rozbicie pojedynczych transakcji na kilka bardziej szczegółowych części. **GnuCash** potrafi także importować i dołączać dane z plików QIF programu **Quicken**. Obsługuje również większość międzynarodowych formatów dat i waluty.

By zainstalować **GnuCash** należy wpisać:

```
# pkg_add -r gnucash
```

Jeśli pakiet nie jest dostępny, możemy wykorzystać kolekcję portów:

```
# cd /usr/ports/finance/gnucash
# make install clean
```

6.5.2. Gnumeric

Gnumeric jest arkuszem kalkulacyjnym, dostępnym jako część środowiska **GNOME**. Dysponuje wygodnym systemem automatycznego „zgadywania” wprowadzanych danych zależnie od formatu komórki oraz automatycznego uzupełniania różnych sekwencji. Potrafi importować pliki z wielu popularnych formatów, jak np. **Excel**, **Lotus 1-2-3** lub **Quattro Pro**. **Gnumeric** pozwala również na kreślenie grafów za pomocą program `math/guppi`. Ponadto, posiada on wiele wbudowanych funkcji oraz wszystkie typowe formaty komórek jak liczby, waluty, daty, czas i wiele innych.

By zainstalować **Gnumeric** z pakietu, należy wpisać:

```
# pkg_add -r gnumeric
```

Jeśli pakiet nie jest dostępny, możemy skorzystać z kolekcji portów:

```
# cd /usr/ports/math/gnumeric
# make install clean
```

6.5.3. Abacus

Abacus jest małym i prostym w użyciu arkuszem kalkulacyjnym. Zawiera on wiele wbudowanych funkcji przydatnych w takich dziedzinach jak statystyka, finanse czy matematyka. Potrafi importować z- i eksportować do formatu plików **Excel**, jak również przygotować pliki PostScript.

By zainstalować **Abacus** z pakietu, należy:

```
# pkg_add -r abacus
```

Jeśli pakiet nie jest dostępny, możemy wykorzystać kolekcję portów:

```
# cd /usr/ports/deskutils/abacus
# make install clean
```

6.6. Podsumowanie

O ile FreeBSD jest popularnym systemem operacyjnym przede wszystkim wśród dostawców usług internetowych, ze względu na swą wydajność i stabilność, o tyle jest on już gotowy do codziennego użytku jako system biurowy. Dzięki dostępności kilku tysięcy aplikacji w postaci pakietów (<http://www.FreeBSD.org/where.html>) bądź portów (<http://www.FreeBSD.org/ports/index.html>), możemy przygotować doskonałe środowisko pracy, w pełni odpowiadające naszym potrzebom.

Mając już zainstalowany system możemy zrobić o jeden krok dalej i wykorzystać `misc/instant-workstation`. Ten „meta-port” pozwala nam skompilować typowy zestaw portów wykorzystywanych w stacjach roboczych. Możemy dopasować go do własnych potrzeb modyfikując plik

`/usr/ports/misc/instant-workstation/Makefile`. Przy dodawaniu i usuwaniu portów należy zachować składnię pliku przedstawioną w domyślnej konfiguracji. Ostatecznie kompilacja przebiega według standardowej procedury. W ten sposób będziemy w stanie przygotować duży pakiet odpowiadający naszemu własnemu środowisku pracy i instalować go na innych stacjach roboczych!

Poniżej znajduje się krótka charakterystyka wszystkich aplikacji biurowych omówionych w tym rozdziale:

Nazwa aplikacji	Nazwa pakietu	Nazwa portu
Mozilla	mozilla	www/mozilla
Opera	opera	www/opera
Firefox	firefox	www/firefox
KOffice	koffice-kde3	editors/koffice-kde3
AbiWord	abiword	editors/abiword
The GIMP	gimp	graphics/gimp
OpenOffice.org	openoffice	editors/openoffice-1.1
Acrobat Reader	acroread	print/acroread7
gv	gv	print/gv
Xpdf	xpdf	graphics/xpdf
GQview	gqview	graphics/gqview
GnuCash	gnucash	finance/gnucash
Gnumeric	gnumeric	math/gnumeric
Abacus	abacus	deskutils/abacus

Rozdział 7. Multimedia

Edited by Ross Lippert.

7.1. Synopsis

FreeBSD supports a wide variety of sound cards, allowing you to enjoy high fidelity output from your computer. This includes the ability to record and playback audio in the MPEG Audio Layer 3 (MP3), WAV, and Ogg Vorbis formats as well as many other formats. The FreeBSD Ports Collection also contains applications allowing you to edit your recorded audio, add sound effects, and control attached MIDI devices.

With some willingness to experiment, FreeBSD can support playback of video files and DVD's. The number of applications to encode, convert, and playback various video media is more limited than the number of sound applications. For example as of this writing, there is no good re-encoding application in the FreeBSD Ports Collection, which could be use to convert between formats, as there is with `audio/sox`. However, the software landscape in this area is changing rapidly.

This chapter will describe the necessary steps to configure your sound card. The configuration and installation of X11 (Rozdział 5) has already taken care of the hardware issues for your video card, though there may be some tweaks to apply for better playback.

After reading this chapter, you will know:

- How to configure your system so that your sound card is recognized.
- Methods to test that your card is working using sample applications.
- How to troubleshoot your sound setup.
- How to playback and encode MP3s and other audio.
- How video is supported by the X server.
- Some video player/encoder ports which give good results.
- How to playback DVD's, `.mpg` and `.avi` files.
- How to rip CD and DVD information into files.
- How to configure a TV card.
- How to configure an image scanner.

Before reading this chapter, you should:

- Know how to configure and install a new kernel (Rozdział 8).

Ostrzeżenie Trying to mount audio CDs with the `mount(8)` command will result in an error, at least, and a *kernel panic*, at worst. These media have specialized encodings which differ from the usual ISO-filesystem.

7.2. Setting Up the Sound Card

Contributed by Moses Moore. Enhanced for FreeBSD 5.X by Marc Fonvieille.

7.2.1. Configuring the System

Before you begin, you should know the model of the card you have, the chip it uses, and whether it is a PCI or ISA card. FreeBSD supports a wide variety of both PCI and ISA cards. Check the supported audio devices list of the Hardware Notes (<http://www.FreeBSD.org/releases/9.1R/hardware.html>) to see if your card is supported. This document will also mention which driver supports your card.

To use your sound device, you will need to load the proper device driver. This may be accomplished in one of two ways. The easiest way is to simply load a kernel module for your sound card with `kldload(8)` which can either be done from the command line:

```
# kldload snd_emu10k1
```

or by adding the appropriate line to the file `/boot/loader.conf` like this:

```
snd_emu10k1_load="YES"
```

These examples are for a Creative SoundBlaster® Live! sound card. Other available loadable sound modules are listed in `/boot/defaults/loader.conf`. If you are not sure which driver to use, you may try to load the `snd_driver` module:

```
# kldload snd_driver
```

This is a metadriver loading the most common device drivers at once. This speeds up the search for the correct driver. It is also possible to load all sound drivers via the `/boot/loader.conf` facility.

If you wish to find out the driver selected for your soundcard after loading the `snd_driver` metadriver, you may check the `/dev/sndstat` file with the `cat /dev/sndstat` command.

A second method is to statically compile in support for your sound card in your kernel. The section below provides the information you need to add support for your hardware in this manner. For more information about recompiling your kernel, please see Rozdział 8.

7.2.1.1. Configuring a Custom Kernel with Sound Support

The first thing to do is adding the generic audio driver `sound(4)` to the kernel, for that you will need to add the following line to the kernel configuration file:

```
device sound
```

Then we have to add the support for our sound card. Therefore, we need to know which driver supports the card. Check the supported audio devices list of the Hardware Notes (<http://www.FreeBSD.org/releases/9.1R/hardware.html>), to determine the correct driver for your sound card. For example, a Creative SoundBlaster Live! sound card is supported by the `snd_emu10k1(4)` driver. To add the support for this card, use the following:

```
device snd_emu10k1
```

Be sure to read the manual page of the driver for the syntax to use. Information regarding the syntax of sound drivers in the kernel configuration can also be found in the `/usr/src/sys/conf/NOTES` file.

Non-PnP ISA cards may require you to provide the kernel with information on the sound card settings (IRQ, I/O port, etc). This is done via the `/boot/device.hints` file. At system boot, the loader(8) will read this file and pass the settings to the kernel. For example, an old Creative SoundBlaster 16 ISA non-PnP card will use the `snd_sbc(4)` driver in conjunction with `snd_sb16(4)`. For this card the following lines have to be added to the kernel configuration file:

```
device snd_sbc
device snd_sb16
```

as well as the following in `/boot/device.hints`:

```
hint.sbc.0.at="isa"
hint.sbc.0.port="0x220"
hint.sbc.0.irq="5"
hint.sbc.0.drq="1"
hint.sbc.0.flags="0x15"
```

In this case, the card uses the 0x220 I/O port and the IRQ 5.

The syntax used in the `/boot/device.hints` file is covered in the sound driver manual page. On FreeBSD 4.X, these settings are directly written in the kernel configuration file.

The settings shown above are the defaults. In some cases, you may need to change the IRQ or the other settings to match your card. See the `snd_sbc(4)` manual page for more information.

7.2.2. Testing the Sound Card

After rebooting with the modified kernel, or after loading the required module, the sound card should appear in your system message buffer (`dmesg(8)`) as something like:

```
pcm0: <Intel ICH3 (82801CA)> port 0xdc80-0xdcbf,0xd800-0xd8ff irq 5 at device 31.5 on pci0
pcm0: [GIANT-LOCKED]
pcm0: <Cirrus Logic CS4205 AC97 Codec>
```

The status of the sound card may be checked via the `/dev/sndstat` file:

```
# cat /dev/sndstat
FreeBSD Audio Driver (newpcm)
Installed devices:
pcm0: <Intel ICH3 (82801CA)> at io 0xd800, 0xdc80 irq 5 bufsz 16384
kld snd_ich (1p/2r/0v channels duplex default)
```

The output from your system may vary. If no `pcm` devices show up, go back and review what was done earlier. Go through your kernel configuration file again and make sure the correct device is chosen. Common problems are listed in Sekcja 7.2.2.1.

If all goes well, you should now have a functioning sound card. If your CD-ROM or DVD-ROM drive is properly coupled to your sound card, you can put a CD in the drive and play it with `cdcontrol(1)`:

```
% cdcontrol -f /dev/acd0 play 1
```

Various applications, such as `audio/workman` can provide a friendlier interface. You may want to install an application such as `audio/mpg123` to listen to MP3 audio files. A quick way to test the card is sending data to the `/dev/dsp`, like this:

```
% cat filename > /dev/dsp
```

where *filename* can be any file. This command line should produce some noise, confirming the sound card is actually working.

Sound card mixer levels can be changed via the `mixer(8)` command. More details can be found in the `mixer(8)` manual page.

7.2.2.1. Common Problems

Error	Solution
unsupported subdevice XX	One or more of the device nodes was not created correctly. Repeat the steps above.
sb_dspwr(XX) timed out	The I/O port is not set correctly.
bad irq XX	The IRQ is set incorrectly. Make sure that the set IRQ and the sound IRQ are the same.
xxx: gus pcm not attached, out of memory	There is not enough available memory to use the device.
xxx: can't open /dev/dsp!	Check with <code>fstat grep dsp</code> if another application is holding the device open. Noteworthy troublemakers are esound and KDE's sound support.

7.2.3. Utilizing Multiple Sound Sources

Contributed by Munish Chopra.

It is often desirable to have multiple sources of sound that are able to play simultaneously, such as when **esound** or **artsd** do not support sharing of the sound device with a certain application.

FreeBSD lets you do this through *Virtual Sound Channels*, which can be set with the `sysctl(8)` facility. Virtual channels allow you to multiplex your sound card's playback channels by mixing sound in the kernel.

To set the number of virtual channels, there are two `sysctl` knobs which, if you are the `root` user, can be set like this:

```
# sysctl hw.snd.pcm0.vchans=4
# sysctl hw.snd.maxautovchans=4
```

The above example allocates four virtual channels, which is a practical number for everyday use.

`hw.snd.pcm0.vchans` is the number of virtual channels `pcm0` has, and is configurable once a device has been attached. `hw.snd.maxautovchans` is the number of virtual channels a new audio device is given when it is attached using `kldload(8)`. Since the `pcm` module can be loaded independently of the hardware drivers,

`hw.snd.maxautovchans` can store how many virtual channels any devices which are attached later will be given.

Notatka: You cannot change the number of virtual channels for a device while it is in use. First close any programs using the device, such as music players or sound daemons.

If you are not using `devfs(5)`, you will have to point your applications at `/dev/dsp0.x`, where `x` is 0 to 3 if `hw.snd.pcm.0.vchans` is set to 4 as in the above example. On a system using `devfs(5)`, the above will automatically be allocated transparently to the user.

7.2.4. Setting Default Values for Mixer Channels

Contributed by Josef El-Rayes.

Notatka: This is only supported in FreeBSD 5.3-RELEASE and later.

The default values for the different mixer channels are hardcoded in the sourcecode of the `pcm(4)` driver. There are a lot of different applications and daemons that allow you to set values for the mixer they remember and set each time they are started, but this is not a clean solution, we want to have default values at the driver level. This is accomplished by defining the appropriate values in `/boot/device.hints`. E.g.:

```
hint.pcm.0.vol="100"
```

This will set the volume channel to a default value of 100, when the `pcm(4)` module is loaded.

7.3. MP3 Audio

Contributed by Chern Lee.

MP3 (MPEG Layer 3 Audio) accomplishes near CD-quality sound, leaving no reason to let your FreeBSD workstation fall short of its offerings.

7.3.1. MP3 Players

By far, the most popular X11 MP3 player is **XMMS** (X Multimedia System). **Winamp** skins can be used with **XMMS** since the GUI is almost identical to that of Nullsoft's **Winamp**. **XMMS** also has native plug-in support.

XMMS can be installed from the `multimedia/xmms` port or package.

XMMS' interface is intuitive, with a playlist, graphic equalizer, and more. Those familiar with **Winamp** will find **XMMS** simple to use.

The `audio/mpg123` port is an alternative, command-line MP3 player.

mpg123 can be run by specifying the sound device and the MP3 file on the command line, as shown below:

```
# mpg123 -a /dev/dsp1.0 FooBar-GreatestHits.mp3
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2 and 3.
Version 0.59r (1999/Jun/15). Written and copyrights by Michael Hipp.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!
```

```
Playing MPEG stream from Foobar-GreatestHits.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
```

`/dev/dsp1.0` should be replaced with the `dsp` device entry on your system.

7.3.2. Ripping CD Audio Tracks

Before encoding a CD or CD track to MP3, the audio data on the CD must be ripped onto the hard drive. This is done by copying the raw CDDA (CD Digital Audio) data to WAV files.

The `cdda2wav` tool, which is a part of the `sysutils/cdrtools` suite, is used for ripping audio information from CDs and the information associated with them.

With the audio CD in the drive, the following command can be issued (as `root`) to rip an entire CD into individual (per track) WAV files:

```
# cdda2wav -D 0,1,0 -B
```

cdda2wav will support ATAPI (IDE) CDROM drives. To rip from an IDE drive, specify the device name in place of the SCSI unit numbers. For example, to rip track 7 from an IDE drive:

```
# cdda2wav -D /dev/acd0a -t 7
```

The `-D 0,1,0` indicates the SCSI device `0,1,0`, which corresponds to the output of `cdrecord -scanbus`.

To rip individual tracks, make use of the `-t` option as shown:

```
# cdda2wav -D 0,1,0 -t 7
```

This example rips track seven of the audio CDROM. To rip a range of tracks, for example, track one to seven, specify a range:

```
# cdda2wav -D 0,1,0 -t 1+7
```

The utility `dd(1)` can also be used to extract audio tracks on ATAPI drives, read Sekcja 17.6.5 for more information on that possibility.

7.3.3. Encoding MP3s

Nowadays, the mp3 encoder of choice is **lame**. **Lame** can be found at `audio/lame` in the ports tree.

Using the ripped WAV files, the following command will convert `audio01.wav` to `audio01.mp3`:

```
# lame -h -b 128 \
--tt "Foo Song Title" \
--ta "FooBar Artist" \
--tl "FooBar Album" \
--ty "2001" \
--tc "Ripped and encoded by Foo" \
--tg "Genre" \
```

```
audio01.wav audio01.mp3
```

128 kbits seems to be the standard MP3 bitrate in use. Many enjoy the higher quality 160, or 192. The higher the bitrate, the more disk space the resulting MP3 will consume--but the quality will be higher. The `-h` option turns on the „higher quality but a little slower” mode. The options beginning with `--t` indicate ID3 tags, which usually contain song information, to be embedded within the MP3 file. Additional encoding options can be found by consulting the lame man page.

7.3.4. Decoding MP3s

In order to burn an audio CD from MP3s, they must be converted to a non-compressed WAV format. Both **XMMS** and **mpg123** support the output of MP3 to an uncompressed file format.

Writing to Disk in **XMMS**:

1. Launch **XMMS**.
2. Right-click on the window to bring up the **XMMS** menu.
3. Select **Preference** under **Options**.
4. Change the Output Plugin to „Disk Writer Plugin”.
5. Press **Configure**.
6. Enter (or choose browse) a directory to write the uncompressed files to.
7. Load the MP3 file into **XMMS** as usual, with volume at 100% and EQ settings turned off.
8. Press **Play** — **XMMS** will appear as if it is playing the MP3, but no music will be heard. It is actually playing the MP3 to a file.
9. Be sure to set the default Output Plugin back to what it was before in order to listen to MP3s again.

Writing to stdout in **mpg123**:

1. Run `mpg123 -s audio01.mp3 > audio01.pcm`

XMMS writes a file in the WAV format, while **mpg123** converts the MP3 into raw PCM audio data. Both of these formats can be used with **cdrecord** to create audio CDs. You have to use raw PCM with `burncd(8)`. If you use WAV files, you will notice a small tick sound at the beginning of each track, this sound is the header of the WAV file. You can simply remove the header of a WAV file with the utility **SoX** (it can be installed from the `audio/sox` port or package):

```
% sox -t wav -r 44100 -s -w -c 2 track.wav track.raw
```

Read Sekcja 17.6 for more information on using a CD burner in FreeBSD.

7.4. Video Playback

Contributed by Ross Lippert.

Video playback is a very new and rapidly developing application area. Be patient. Not everything is going to work as smoothly as it did with sound.

Before you begin, you should know the model of the video card you have and the chip it uses. While **Xorg** and **XFree86** support a wide variety of video cards, fewer give good playback performance. To obtain a list of extensions supported by the X server using your card use the command `xdpyinfo(1)` while X11 is running.

It is a good idea to have a short MPEG file which can be treated as a test file for evaluating various players and options. Since some DVD players will look for DVD media in `/dev/dvd` by default, or have this device name hardcoded in them, you might find it useful to make symbolic links to the proper devices:

```
# ln -sf /dev/acd0c /dev/dvd
# ln -sf /dev/racd0c /dev/rdvd
```

On FreeBSD 5.X, which uses `devfs(5)` there is a slightly different set of recommended links:

```
# ln -sf /dev/acd0 /dev/dvd
# ln -sf /dev/acd0 /dev/rdvd
```

Note that due to the nature of `devfs(5)`, manually created links like these will not persist if you reboot your system. In order to create the symbolic links automatically whenever you boot your system, add the following lines to `/etc/devfs.conf`:

```
link acd0 dvd
link acd0 rdvd
```

Additionally, DVD decryption, which requires invoking special DVD-ROM functions, requires write permission on the DVD devices.

Some of the ports discussed rely on the following kernel options to build correctly. Before attempting to build, add this option to the kernel configuration file, build a new kernel, and reboot:

```
options CPU_ENABLE_SSE
```

To enhance the shared memory X11 interface, it is recommended that the values of some `sysctl(8)` variables should be increased:

```
kern.ipc.shmmax=67108864
kern.ipc.shmall=32768
```

7.4.1. Determining Video Capabilities

There are several possible ways to display video under X11. What will really work is largely hardware dependent. Each method described below will have varying quality across different hardware. Secondly, the rendering of video in X11 is a topic receiving a lot of attention lately, and with each version of **Xorg**, or of **XFree86**, there may be significant improvement.

A list of common video interfaces:

1. X11: normal X11 output using shared memory.

2. XVideo: an extension to the X11 interface which supports video in any X11 drawable.
3. SDL: the Simple Directmedia Layer.
4. DGA: the Direct Graphics Access.
5. SVGAlib: low level console graphics layer.

7.4.1.1. XVideo

Xorg and **XFree86 4.X** have an extension called *XVideo* (aka Xvideo, aka Xv, aka xv) which allows video to be directly displayed in drawable objects through a special acceleration. This extension provides very good quality playback even on low-end machines.

To check whether the extension is running, use `xvinfo`:

```
% xvinfo
```

XVideo is supported for your card if the result looks like:

```
X-Video Extension version 2.2
screen #0
  Adaptor #0: "Savage Streams Engine"
    number of ports: 1
    port base: 43
    operations supported: PutImage
    supported visuals:
      depth 16, visualID 0x22
      depth 16, visualID 0x23
    number of attributes: 5
      "XV_COLORKEY" (range 0 to 16777215)
        client settable attribute
        client gettable attribute (current value is 2110)
      "XV_BRIGHTNESS" (range -128 to 127)
        client settable attribute
        client gettable attribute (current value is 0)
      "XV_CONTRAST" (range 0 to 255)
        client settable attribute
        client gettable attribute (current value is 128)
      "XV_SATURATION" (range 0 to 255)
        client settable attribute
        client gettable attribute (current value is 128)
      "XV_HUE" (range -180 to 180)
        client settable attribute
        client gettable attribute (current value is 0)
  maximum XvImage size: 1024 x 1024
  Number of image formats: 7
    id: 0x32595559 (YUY2)
      guid: 59555932-0000-0010-8000-00aa00389b71
      bits per pixel: 16
      number of planes: 1
      type: YUV (packed)
    id: 0x32315659 (YV12)
      guid: 59563132-0000-0010-8000-00aa00389b71
      bits per pixel: 12
```

```

    number of planes: 3
    type: YUV (planar)
id: 0x30323449 (I420)
    guid: 49343230-0000-0010-8000-00aa00389b71
    bits per pixel: 12
    number of planes: 3
    type: YUV (planar)
id: 0x36315652 (RV16)
    guid: 52563135-0000-0000-0000-000000000000
    bits per pixel: 16
    number of planes: 1
    type: RGB (packed)
    depth: 0
    red, green, blue masks: 0x1f, 0x3e0, 0x7c00
id: 0x35315652 (RV15)
    guid: 52563136-0000-0000-0000-000000000000
    bits per pixel: 16
    number of planes: 1
    type: RGB (packed)
    depth: 0
    red, green, blue masks: 0x1f, 0x7e0, 0xf800
id: 0x31313259 (Y211)
    guid: 59323131-0000-0010-8000-00aa00389b71
    bits per pixel: 6
    number of planes: 3
    type: YUV (packed)
id: 0x0
    guid: 00000000-0000-0000-0000-000000000000
    bits per pixel: 0
    number of planes: 0
    type: RGB (packed)
    depth: 1
    red, green, blue masks: 0x0, 0x0, 0x0

```

Also note that the formats listed (YUV2, YUV12, etc) are not present with every implementation of XVideo and their absence may hinder some players.

If the result looks like:

```

X-Video Extension version 2.2
screen #0
no adaptors present

```

Then XVideo is probably not supported for your card.

If XVideo is not supported for your card, this only means that it will be more difficult for your display to meet the computational demands of rendering video. Depending on your video card and processor, though, you might still be able to have a satisfying experience. You should probably read about ways of improving performance in the advanced reading Sekcja 7.4.3.

7.4.1.2. Simple Directmedia Layer

The Simple Directmedia Layer, SDL, was intended to be a porting layer between Microsoft Windows, BeOS, and UNIX, allowing cross-platform applications to be developed which made efficient use of sound and graphics. The SDL layer provides a low-level abstraction to the hardware which can sometimes be more efficient than the X11 interface.

The SDL can be found at `devel/sdl12`.

7.4.1.3. Direct Graphics Access

Direct Graphics Access is an X11 extension which allows a program to bypass the X server and directly alter the framebuffer. Because it relies on a low level memory mapping to effect this sharing, programs using it must be run as `root`.

The DGA extension can be tested and benchmarked by `dga(1)`. When `dga` is running, it changes the colors of the display whenever a key is pressed. To quit, use `q`.

7.4.2. Ports and Packages Dealing with Video

This section discusses the software available from the FreeBSD Ports Collection which can be used for video playback. Video playback is a very active area of software development, and the capabilities of various applications are bound to diverge somewhat from the descriptions given here.

Firstly, it is important to know that many of the video applications which run on FreeBSD were developed as Linux applications. Many of these applications are still beta-quality. Some of the problems that you may encounter with video packages on FreeBSD include:

1. An application cannot playback a file which another application produced.
2. An application cannot playback a file which the application itself produced.
3. The same application on two different machines, rebuilt on each machine for that machine, plays back the same file differently.
4. A seemingly trivial filter like rescaling of the image size results in very bad artifacts from a buggy rescaling routine.
5. An application frequently dumps core.
6. Documentation is not installed with the port and can be found either on the web or under the port's `work` directory.

Many of these applications may also exhibit „Linux-isms”. That is, there may be issues resulting from the way some standard libraries are implemented in the Linux distributions, or some features of the Linux kernel which have been assumed by the authors of the applications. These issues are not always noticed and worked around by the port maintainers, which can lead to problems like these:

1. The use of `/proc/cpuinfo` to detect processor characteristics.
2. A misuse of threads which causes a program to hang upon completion instead of truly terminating.
3. Software not yet in the FreeBSD Ports Collection which is commonly used in conjunction with the application.

So far, these application developers have been cooperative with port maintainers to minimize the work-arounds needed for port-ing.

7.4.2.1. MPlayer

MPlayer is a recently developed and rapidly developing video player. The goals of the **MPlayer** team are speed and flexibility on Linux and other Unices. The project was started when the team founder got fed up with bad playback performance on then available players. Some would say that the graphical interface has been sacrificed for a streamlined design. However, once you get used to the command line options and the key-stroke controls, it works very well.

7.4.2.1.1. Building MPlayer

MPlayer resides in `multimedia/mplayer`. **MPlayer** performs a variety of hardware checks during the build process, resulting in a binary which will not be portable from one system to another. Therefore, it is important to build it from ports and not to use a binary package. Additionally, a number of options can be specified in the make command line, as described in the `Makefile` and at the start of the build:

```
# cd /usr/ports/multimedia/mplayer
# make
N - O - T - E
```

```
Take a careful look into the Makefile in order
to learn how to tune mplayer towards you personal preferences!
For example,
make WITH_GTK1
builds MPlayer with GTK1-GUI support.
If you want to use the GUI, you can either install
/usr/ports/multimedia/mplayer-skins
or download official skin collections from
http://www.mplayerhq.hu/homepage/dload.html
```

The default port options should be sufficient for most users. However, if you need the XviD codec, you have to specify the `WITH_XVID` option in the command line. The default DVD device can also be defined with the `WITH_DVD_DEVICE` option, by default `/dev/acd0` will be used.

As of this writing, the **MPlayer** port will build its HTML documentation and two executables, `mplayer`, and `mencoder`, which is a tool for re-encoding video.

The HTML documentation for **MPlayer** is very informative. If the reader finds the information on video hardware and interfaces in this chapter lacking, the **MPlayer** documentation is a very thorough supplement. You should definitely take the time to read the **MPlayer** documentation if you are looking for information about video support in UNIX.

7.4.2.1.2. Using MPlayer

Any user of **MPlayer** must set up a `.mplayer` subdirectory of her home directory. To create this necessary subdirectory, you can type the following:

```
% cd /usr/ports/multimedia/mplayer
% make install-user
```

The command options for `mplayer` are listed in the manual page. For even more detail there is HTML documentation. In this section, we will describe only a few common uses.

To play a file, such as `testfile.avi`, through one of the various video interfaces set the `-vo` option:

```
% mplayer -vo xv testfile.avi
% mplayer -vo sdl testfile.avi
% mplayer -vo x11 testfile.avi
# mplayer -vo dga testfile.avi
# mplayer -vo 'sdl:dga' testfile.avi
```

It is worth trying all of these options, as their relative performance depends on many factors and will vary significantly with hardware.

To play from a DVD, replace the `testfile.avi` with `dvd://N -dvd-device DEVICE` where `N` is the title number to play and `DEVICE` is the device node for the DVD-ROM. For example, to play title 3 from `/dev/dvd`:

```
# mplayer -vo xv dvd://3 -dvd-device /dev/dvd
```

Notatka: The default DVD device can be defined during the build of the **MPlayer** port via the `WITH_DVD_DEVICE` option. By default, this device is `/dev/acd0`. More details can be found in the port `Makefile`.

To stop, pause, advance and so on, consult the keybindings, which are output by running `mplayer -h` or read the manual page.

Additional important options for playback are: `-fs -zoom` which engages the fullscreen mode and `-framedrop` which helps performance.

In order for the `mplayer` command line to not become too large, the user can create a file `.mplayer/config` and set default options there:

```
vo=xv
fs=yes
zoom=yes
```

Finally, `mplayer` can be used to rip a DVD title into a `.vob` file. To dump out the second title from a DVD, type this:

```
# mplayer -dumpstream -dumpfile out.vob dvd://2 -dvd-device /dev/dvd
```

The output file, `out.vob`, will be MPEG and can be manipulated by the other packages described in this section.

7.4.2.1.3. *mencoder*

Before using `mencoder` it is a good idea to familiarize yourself with the options from the HTML documentation. There is a manual page, but it is not very useful without the HTML documentation. There are innumerable ways to improve quality, lower bitrate, and change formats, and some of these tricks may make the difference between good or bad performance. Here are a couple of examples to get you going. First a simple copy:

```
% mencoder input.avi -oac copy -ovc copy -o output.avi
```

Improper combinations of command line options can yield output files that are unplayable even by `mplayer`. Thus, if you just want to rip to a file, stick to the `-dumpfile` in `mplayer`.

To convert `input.avi` to the MPEG4 codec with MPEG3 audio encoding (audio/lame is required):

```
% mencoder input.avi -oac mp3lame -lameopts br=192 \
  -ovc lavc -lavcopts vcodec=mpeg4:vhq -o output.avi
```

This has produced output playable by `mplayer` and `xine`.

`input.avi` can be replaced with `dvd://1 -dvd-device /dev/dvd` and run as `root` to re-encode a DVD title directly. Since you are likely to be dissatisfied with your results the first time around, it is recommended you dump the title to a file and work on the file.

7.4.2.2. The `xine` Video Player

The **`xine`** video player is a project of wide scope aiming not only at being an all in one video solution, but also in producing a reusable base library and a modular executable which can be extended with plugins. It comes both as a package and as a port, `multimedia/xine`.

The **`xine`** player is still very rough around the edges, but it is clearly off to a good start. In practice, **`xine`** requires either a fast CPU with a fast video card, or support for the XVideo extension. The GUI is usable, but a bit clumsy.

As of this writing, there is no input module shipped with **`xine`** which will play CSS encoded DVD's. There are third party builds which do have modules for this built in them, but none of these are in the FreeBSD Ports Collection.

Compared to **`MPlayer`**, **`xine`** does more for the user, but at the same time, takes some of the more fine-grained control away from the user. The **`xine`** video player performs best on XVideo interfaces.

By default, **`xine`** player will start up in a graphical user interface. The menus can then be used to open a specific file:

```
% xine
```

Alternatively, it may be invoked to play a file immediately without the GUI with the command:

```
% xine -g -p mymovie.avi
```

7.4.2.3. The `transcode` Utilities

The software **`transcode`** is not a player, but a suite of tools for re-encoding video and audio files. With **`transcode`**, one has the ability to merge video files, repair broken files, using command line tools with `stdin/stdout` stream interfaces.

A great number of options can be specified during the build from the `multimedia/transcode` port, we recommend the following command line to build **`transcode`**:

```
# make WITH_OPTIMIZED_CFLAGS=yes WITH_LIBA52=yes WITH_LAME=yes WITH_OGG=yes \
  WITH_MJPEG=yes -DWITH_XVID=yes
```

The proposed settings should be sufficient for most users.

To illustrate transcode capacities, one example to show how to convert a DivX file into a PAL MPEG-1 file (PAL VCD):

```
% transcode -i input.avi -V --export_prof vcd-pal -o output_vcd
% mplex -f 1 -o output_vcd.mpg output_vcd.mlv output_vcd.mpa
```

The resulting MPEG file, `output_vcd.mpg`, is ready to be played with **MPlayer**. You could even burn the file on a CD-R media to create a Video CD, in this case you will need to install and use both `multimedia/vcdimager` and `sysutils/cdrdao` programs.

There is a manual page for `transcode`, but you should also consult the `transcode` wiki (<http://www.transcoding.org/cgi-bin/transcode>) for further information and examples.

7.4.3. Further Reading

The various video software packages for FreeBSD are developing rapidly. It is quite possible that in the near future many of the problems discussed here will have been resolved. In the mean time, those who want to get the very most out of FreeBSD's A/V capabilities will have to cobble together knowledge from several FAQs and tutorials and use a few different applications. This section exists to give the reader pointers to such additional information.

The MPlayer documentation (<http://www.mplayerhq.hu/DOCS/>) is very technically informative. These documents should probably be consulted by anyone wishing to obtain a high level of expertise with UNIX video. The **MPlayer** mailing list is hostile to anyone who has not bothered to read the documentation, so if you plan on making bug reports to them, RTFM.

The `xine` HOWTO (http://dvd.sourceforge.net/xine-howto/en_GB/html/howto.html) contains a chapter on performance improvement which is general to all players.

Finally, there are some other promising applications which the reader may try:

- `Avifile` (<http://avifile.sourceforge.net/>) which is also a port `multimedia/avifile`.
- `Ogle` (<http://www.dtek.chalmers.se/groups/dvd/>) which is also a port `multimedia/ogle`.
- `Xtheater` (<http://xtheater.sourceforge.net/>)
- `multimedia/dvdauthor`, an open source package for authoring DVD content.

7.5. Setting Up TV Cards

Original contribution by Josef El-Rayes. Enhanced and adapted by Marc Fonvieille.

7.5.1. Introduction

TV cards allow you to watch broadcast or cable TV on your computer. Most of them accept composite video via an RCA or S-video input and some of these cards come with a FM radio tuner.

FreeBSD provides support for PCI-based TV cards using a Brooktree Bt848/849/878/879 or a Conexant CN-878/Fusion 878a Video Capture Chip with the `bktr(4)` driver. You must also ensure the board comes with a supported tuner, consult the `bktr(4)` manual page for a list of supported tuners.

7.5.2. Adding the Driver

To use your card, you will need to load the `bktr(4)` driver, this can be done by adding the following line to the `/boot/loader.conf` file like this:

```
bktr_load="YES"
```

Alternatively, you may statically compile the support for the TV card in your kernel, in that case add the following lines to your kernel configuration:

```
device    bktr
device    iicbus
device    iicbb
device    smbus
```

These additional device drivers are necessary because of the card components being interconnected via an I2C bus. Then build and install a new kernel.

Once the support was added to your system, you have to reboot your machine. During the boot process, your TV card should show up, like this:

```
bktr0: <BrookTree 848A> mem 0xd7000000-0xd7000fff irq 10 at device 10.0 on pci0
iicbb0: <I2C bit-banging driver> on bti2c0
iicbus0: <Philips I2C bus> on iicbb0 master-only
iicbus1: <Philips I2C bus> on iicbb0 master-only
smbus0: <System Management Bus> on bti2c0
bktr0: Pinnacle/Miro TV, Philips SECAM tuner.
```

Of course these messages can differ according to your hardware. However you should check if the tuner is correctly detected; it is still possible to override some of the detected parameters with `sysctl(8)` MIBs and kernel configuration file options. For example, if you want to force the tuner to a Philips SECAM tuner, you should add the following line to your kernel configuration file:

```
options OVERRIDE_TUNER=6
```

or you can directly use `sysctl(8)`:

```
# sysctl hw.bt848.tuner=6
```

See the `bktr(4)` manual page and the `/usr/src/sys/conf/NOTES` file for more details on the available options.

7.5.3. Useful Applications

To use your TV card you need to install one of the following applications:

- `multimedia/fxrtv` provides TV-in-a-window and image/audio/video capture capabilities.
- `multimedia/xawtv` is also a TV application, with the same features as `fxrtv`.
- `misc/alevt` decodes and displays Videotext/Teletext.
- `audio/xmradio`, an application to use the FM radio tuner coming with some TV cards.
- `audio/wmtune`, a handy desktop application for radio tuners.

More applications are available in the FreeBSD Ports Collection.

7.5.4. Troubleshooting

If you encounter any problem with your TV card, you should check at first if the video capture chip and the tuner are really supported by the bktr(4) driver and if you used the right configuration options. For more support and various questions about your TV card you may want to contact and use the archives of the freebsd-multimedia (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-multimedia>) mailing list.

7.6. Image Scanners

Written by Marc Fonvieille.

7.6.1. Introduction

FreeBSD, like any modern operating system, allows the use of image scanners. Standardized access to scanners is provided by the **SANE** (Scanner Access Now Easy) API available through the FreeBSD Ports Collection. **SANE** will also use some FreeBSD device drivers to access the scanner hardware.

FreeBSD supports both SCSI and USB scanners. Be sure your scanner is supported by **SANE** prior to performing any configuration. **SANE** has a supported devices (<http://sane-project.org/sane-supported-devices.html>) list that can provide you with information about the support for a scanner and its status. The usscanner(4) manual page also provides a list of supported USB scanners.

7.6.2. Kernel Configuration

As mentioned above both SCSI and USB interfaces are supported. According to your scanner interface, different device drivers are required.

7.6.2.1. USB Interface

The **GENERIC** kernel by default includes the device drivers needed to support USB scanners. Should you decide to use a custom kernel, be sure that the following lines are present in your kernel configuration file:

```
device usb
device uhci
device ohci
device usscanner
```

Depending upon the USB chipset on your motherboard, you will only need either `device uhci` or `device ohci`, however having both in the kernel configuration file is harmless.

If you do not want to rebuild your kernel and your kernel is not the **GENERIC** one, you can directly load the usscanner(4) device driver module with the `kldload(8)` command:

```
# kldload usscanner
```

To load this module at each system startup, add the following line to `/boot/loader.conf`:

```
usscanner_load="YES"
```

After rebooting with the correct kernel, or after loading the required module, plug in your USB scanner. The scanner should appear in your system message buffer (dmesg(8)) as something like:

```
usscanner0: EPSON EPSON Scanner, rev 1.10/3.02, addr 2
```

This shows that our scanner is using the /dev/usscanner0 device node.

7.6.2.2. SCSI Interface

If your scanner comes with a SCSI interface, it is important to know which SCSI controller board you will use. According to the SCSI chipset used, you will have to tune your kernel configuration file. The `GENERIC` kernel supports the most common SCSI controllers. Be sure to read the `NOTES` file and add the correct line to your kernel configuration file. In addition to the SCSI adapter driver, you need to have the following lines in your kernel configuration file:

```
device scbus
device pass
```

Once your kernel has been properly compiled, you should be able to see the devices in your system message buffer, when booting:

```
pass2 at aic0 bus 0 target 2 lun 0
pass2: <AGFA SNAPSCAN 600 1.10> Fixed Scanner SCSI-2 device
pass2: 3.300MB/s transfers
```

If your scanner was not powered-on at system boot, it is still possible to manually force the detection by performing a SCSI bus scan with the `camcontrol(8)` command:

```
# camcontrol rescan all
Re-scan of bus 0 was successful
Re-scan of bus 1 was successful
Re-scan of bus 2 was successful
Re-scan of bus 3 was successful
```

Then the scanner will appear in the SCSI devices list:

```
# camcontrol devlist
<IBM DDRS-34560 S97B>          at scbus0 target 5 lun 0 (pass0,da0)
<IBM DDRS-34560 S97B>          at scbus0 target 6 lun 0 (pass1,da1)
<AGFA SNAPSCAN 600 1.10>      at scbus1 target 2 lun 0 (pass3)
<PHILIPS CDD3610 CD-R/RW 1.00> at scbus2 target 0 lun 0 (pass2,cd0)
```

More details about SCSI devices, are available in the `scsi(4)` and `camcontrol(8)` manual pages.

7.6.3. SANE Configuration

The **SANE** system has been splitted in two parts: the backends (`graphics/sane-backends`) and the frontends (`graphics/sane-frontends`). The backends part provides access to the scanner itself. The **SANE**'s supported

devices (<http://sane-project.org/sane-supported-devices.html>) list specifies which backend will support your image scanner. It is mandatory to determine the correct backend for your scanner if you want to be able to use your device. The frontends part provides the graphical scanning interface (**xscanimage**).

The first thing to do is install the `graphics/sane-backends` port or package. Then, use the `sane-find-scanner` command to check the scanner detection by the **SANE** system:

```
# sane-find-scanner -q
found SCSI scanner "AGFA SNAPSCAN 600 1.10" at /dev/pass3
```

The output will show the interface type of the scanner and the device node used to attach the scanner to the system. The vendor and the product model may not appear, it is not important.

Notatka: Some USB scanners require you to load a firmware, this is explained in the backend manual page. You should also read `sane-find-scanner(1)` and `sane(7)` manual pages.

Now we have to check if the scanner will be identified by a scanning frontend. By default, the **SANE** backends comes with a command line tool called `scanimage(1)`. This command allows you to list the devices and to perform an image acquisition from the command line. The `-L` option is used to list the scanner device:

```
# scanimage -L
device 'snapscan:/dev/pass3' is a AGFA SNAPSCAN 600 flatbed scanner
```

No output or a message saying that no scanners were identified indicates that `scanimage(1)` is unable to identify the scanner. If this happens, you will need to edit the backend configuration file and define the scanner device used. The `/usr/local/etc/sane.d/` directory contains all backends configuration files. This identification problem does appear with certain USB scanners.

For example, with the USB scanner used in the Sekcja 7.6.2.1, `sane-find-scanner` gives us the following information:

```
# sane-find-scanner -q
found USB scanner (UNKNOWN vendor and product) at device /dev/usb/lp0
```

The scanner is correctly detected, it uses the USB interface and is attached to the `/dev/usb/lp0` device node. We can now check if the scanner is correctly identified:

```
# scanimage -L
```

```
No scanners were identified. If you were expecting something different,
check that the scanner is plugged in, turned on and detected by the
sane-find-scanner tool (if appropriate). Please read the documentation
which came with this software (README, FAQ, manpages).
```

Since the scanner is not identified, we will need to edit the `/usr/local/etc/sane.d/epson.conf` file. The scanner model used was the EPSON Perfection® 1650, so we know the scanner will use the `epson` backend. Be sure to read the help comments in the backends configuration files. Line changes are quite simple: comment out all lines that have the wrong interface for your scanner (in our case, we will comment out all lines starting with the word `scsi` as our scanner uses the USB interface), then add at the end of the file a line specifying the interface and the device node used. In this case, we add the following line:

```
usb /dev/usbscanner0
```

Please be sure to read the comments provided in the backend configuration file as well as the backend manual page for more details and correct syntax to use. We can now verify if the scanner is identified:

```
# scanimage -L
device 'epson:/dev/usbscanner0' is a Epson GT-8200 flatbed scanner
```

Our USB scanner has been identified. It is not important if the brand and the model do not match. The key item to be concerned with is the `'epson:/dev/usbscanner0'` field, which give us the right backend name and the right device node.

Once the `scanimage -L` command is able to see the scanner, the configuration is complete. The device is now ready to scan.

While `scanimage(1)` does allow us to perform an image acquisition from the command line, it is preferable to use a graphical user interface to perform image scanning. **SANE** offers a simple but efficient graphical interface:

xscanimage (`graphics/sane-frontends`).

Xsane (`graphics/xsane`) is another popular graphical scanning frontend. This frontend offers advanced features such as various scanning mode (photocopy, fax, etc.), color correction, batch scans, etc. Both of these applications are useable as a **GIMP** plugin.

7.6.4. Allowing Scanner Access to Other Users

All previous operations have been done with `root` privileges. You may however, need other users to have access to the scanner. The user will need read and write permissions to the device node used by the scanner. As an example, our USB scanner uses the device node `/dev/usbscanner0` which is owned by the `operator` group. Adding the user `joe` to the `operator` group will allow him to use the scanner:

```
# pw groupmod operator -m joe
```

For more details read the `pw(8)` manual page. You also have to set the correct write permissions (0660 or 0664) on the `/dev/usbscanner0` device node, by default the `operator` group can only read the device node. This is done by adding the following lines to the `/etc/devfs.rules` file:

```
[system=5]
add path usbscanner0 mode 660
```

Then add the following to `/etc/rc.conf` and reboot the machine:

```
devfs_system_ruleset="system"
```

More information regarding these lines can be found in the `devfs(8)` manual page.

Notatka: Of course, for security reasons, you should think twice before adding a user to any group, especially the `operator` group.

Rozdział 8. Konfiguracja jądra FreeBSD

Zaktualizował i zrekonstruował Jim Mock. Pierwotnie napisał Jake Hamby. Tłumaczył Łukasz Piechowiak.

8.1. Streszczenie

Rdzeniem systemu operacyjnego FreeBSD jest jądro. Odpowiedzialne jest za zarządzanie pamięcią, wymuszanie kontroli bezpieczeństwa, sieć, dostęp do dysków i wiele innych. Podczas, gdy coraz więcej elementów FreeBSD jest konfigurowanych dynamicznie, czasem jeszcze może zająć potrzeba przekonfigurowania i rekompilowania jądra.

Po przeczytaniu tego rozdziału będziemy wiedzieć:

- Dlaczego możemy potrzebować indywidualnego jądra.
- Jak napisać plik konfiguracyjny lub dostroić istniejący.
- Jak wykorzystać plik konfiguracyjny jądra do przygotowania i kompilacji nowego jądra.
- Jak zainstalować nowe jądro.
- Jak się ratować, jeśli coś pójdzie nie tak.

Wszystkie przykładowe polecenia przedstawione w niniejszym rozdziale powinny być uruchamiane jako użytkownik `root`.

8.2. Po co budować indywidualne jądro?

Tradycyjnie, system FreeBSD miał coś, co zwie się „monolitycznym” jądrem. Był to jeden duży program, wspierający ustaloną liczbę urządzeń. Jeśli zaszła potrzeba zmiany zachowania jądra, należało skompilować nowe jądro i uruchomić z nim ponownie komputer.

W dzisiejszych czasach, FreeBSD bardzo szybko przechodzi do modelu, w którym funkcjonalność jądra zawiera się w modułach, które można dynamicznie aplikować, lub usuwać, w miarę potrzeb. Umożliwia to jądrze szybkie przystosowywanie się zaraz po rozpoznaniu nowego sprzętu (jak karty PCMCIA w laptopach). Pozwala też zwiększyć funkcjonalność, której nie miało oryginalne jądro (któremu nie były dane funkcje potrzebne). Potocznie mówi się o jądrze modułarnym.

Pomimo tego, czasem trzeba wprowadzić do jądra statyczne zmiany. Na przykład w sytuacjach, gdy kluczowe funkcje jądra zostają zmieniane, nie jest możliwym załadowanie dynamicznie ładowalnego modułu. Możliwe też, że jeszcze odpowiedni, dynamicznie ładowalny moduł, nie został napisany.

Budowanie indywidualnego jądra jest jednym z najważniejszych rytuałów, których podczas użytkowania systemu BSD trzeba doświadczyć. Ten czasochłonny proces przyniesie naszemu systemowi wiele korzyści. Inaczej niż w przypadku jądra `GENERIC` [podstawowego, domyślnego], które musi wspierać wiele rodzajów sprzętu, nasze jądro będzie wspierało tylko *nasz* sprzęt PC. Ma to wiele zalet:

- Szybszy czas uruchamiania systemu. Od kiedy jądro będzie sprawdzało tylko sprzęt który mamy, czas uruchamiania znacząco się zmniejszy.

- Mniejsze zużycie pamięci. Indywidualne jądro często zużywa mniej pamięci niż jądro `GENERIC`, co jest istotnym faktem, gdyż jądro przez cały czas musi być w pamięci obecne. Z tych powodów, budowanie indywidualnego jądra jest szczególnie przydatne przy pracy z maszynami o małej ilości pamięci RAM'u.
- Więcej wspieranego sprzętu. Indywidualne jądro może zawierać obsługę np. kart muzycznych, które nie są wspierane przez domyślne jądro `GENERIC`.

8.3. Budowanie i instalowanie indywidualnego jądra

Omówmy pokrótce katalog kompilacji jądra. Wszystkie wspomniane za chwilę katalogi będą relatywnymi względem `/usr/src/sys`, do którego można także dojść przez `/sys`. Można tam znaleźć wiele różnych podkatalogów, jednak dla nas najważniejszym będzie `arch/conf`. W nim właśnie dokonamy edycji pliku konfiguracyjnego jądra oraz je skompilujemy, będą to kolejne etapy w całym procesie budowy. `arch` oznacza architekturę, do wyboru: `i386`, `alpha`, `amd64`, `ia64`, `powerpc`, `sparc64`, lub `pc98` (alternatywna gałąź sprzętu PC, popularna w Japonii). Wszystko, co znajduje się w katalogu danej architektury dotyczy ściśle tylko jej. Reszta źródeł jest dla wszystkich architektur taka sama. Zwróćmy uwagę na logiczną strukturę katalogów z każdym wspieranym urządzeniem, systemem plików, opcjami dodatkowymi - wszystko posiada swój własny podkatalog.

Przykłady w niniejszym rozdziale zakładają, że wykorzystujemy architekturę `i386`. Jeśli tak nie jest, będziemy musieli dokonać odpowiednich zmian w nazwach ścieżek dostępu dla architektury naszego systemu.

Notatka: Jeśli *nie* mamy katalogu `/usr/src/sys`, oznacza to, że nie dysponujemy zainstalowanymi źródłami jądra. Najprostszym sposobem na zainstalowanie jest uruchomienie jako `root` `sysinstall'a`, wybranie `Configure`, następnie `Distributions`, później `src`, a na końcu `sys`. Jeśli jednak jesteśmy osobami mającymi awersję do konfiguratorów możemy zainstalować źródła jądra ręcznie. W poniższym przykładzie instalacja z „oficjalnej” płyty CD FreeBSD:

```
# mount /cdrom
# mkdir -p /usr/src/sys
# ln -s /usr/src/sys /sys
# cat /cdrom/src/sys.[a-d]* | tar -xzvf -
```

Następnie wchodzimy do katalogu `arch/conf` i kopiujemy domyślny plik konfiguracyjny o nazwie `GENERIC` tworząc plik z nazwą jaką chcemy nadać swojemu jądru. Na przykład:

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYKERNEL
```

Tradycyjnie nazwa jądra pisana jest wielkimi literami. Dodatkowo dobrym pomysłem jest, by nazywać jądra tak jak komputery, co pomaga rozróżnić jądra, gdy mamy wiele komputerów z różnym sprzętem. Dla potrzeb tego przykładu nazwiemy jądro `MYKERNEL`.

Podpowiedź: Nie jest najlepszym pomysłem trzymanie pliku konfiguracyjnego jądra bezpośrednio w katalogu `/usr/src`. Jeśli podczas kompilacji mamy kłopot, czasem może się okazać kuszącym pomysłem po prostu wykasować cały katalog `/usr/src` i rozpocząć od początku. Wtedy zwykle, kilka sekund po usunięciu katalogu, przypomina nam się, że usunęliśmy także plik konfiguracyjny jądra. Podobnie, nie powinniśmy edytować bezpośrednio `GENERIC`, gdyż może zostać nadpisany przy kolejnej aktualizacji naszego drzewa źródeł i zmiany, które wprowadziliśmy zostaną utracone.

Możemy chcieć trzymać plik konfiguracyjny jądra gdziekolwiek, a następnie utworzyć symboliczne dowiązanie do pliku w katalogu `i386`.

Przykładowo:

```
# cd /usr/src/sys/i386/conf
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL
```

Przyszedł czas na edycję pliku konfiguracyjnego jądra. W przykładzie nazywa się on `MYKERNEL`. Jeśli dopiero zainstalowaliśmy system, jedynym z dostępnych edytorów może być `vi`. Mimo, że jest dobrze udokumentowany, opisany w wielu książkach, dla początkujących wydaje się on nieco zbyt skomplikowany. FreeBSD zaopatrzony jest również w drugi edytor, znacznie prostszy w obsłudze, o nazwie `ee`. Jeśli dopiero zaczynamy, `ee` powinien być naszym wyborem. Nie krępujmy się i zmienmy wartości na gorze pliku, szczególnie te, odróżniające nasz własny plik od `GENERIC`.

Jeśli już kompilowaliśmy jądro w SunOS lub innych systemach BSD, duża część pliku konfiguracyjnego powinna być nam znajoma. Jeśli natomiast jesteśmy lepiej zaznajomieni z systemami typu DOS, plik konfiguracyjny może wydać się nam nieco obcy. W tym przypadku przeczytajmy uważnie każdą opcję oraz komentarz w pliku konfiguracyjnym.

Notatka: Jeśli synchronizujemy nasze drzewo źródłowe z najnowszymi źródłami projektu FreeBSD, należy zawsze, nim rozpoczniemy jakiegokolwiek działania aktualizujące, zapoznać się z zawartością pliku `/usr/src/UPDATING`. W pliku tym zapisane są wszelkie niezbędne zagadnienia związane z aktualizacją FreeBSD. Plik `/usr/src/UPDATING` zawsze pasuje do źródła naszej wersji FreeBSD, jest przez to bardziej odpowiednim źródłem informacji niż Podręcznik.

Musimy teraz skompilować kod źródłowy jądra. Istnieją dwie procedury, za pomocą których można tego dokonać. Wybór zależeć będzie od tego w jakim celu kompilujemy jądro oraz od wykorzystywanej wersji FreeBSD.

- Jeśli zainstalowaliśmy *tylko* źródła jądra, wykorzystamy procedurę 1.
- Jeśli budujemy nowe jądro, bez aktualizowania źródeł (na przykład, by dodać dodatkowe opcje, np. `IPFIREWALL`), możemy użyć dowolnej z procedur.
- Jeśli przebudowujemy jądro jako część procesu `make buildworld`, powinniśmy użyć procedury 2.

Jeśli *nie* aktualizowaliśmy naszych źródeł w żaden sposób od ostatniego, zakończonego powodzeniem cyklu `buildworld-installworld` (nie uruchamialiśmy **CVSup**, **CTM**, ani nie korzystaliśmy z **anoncvs**), wówczas bezpiecznym jest skorzystać z sekwencji `config, make depend, make i make install`.

Procedura 1. Budowanie jądra w „tradycyjny” sposób.

1. By wygenerować kod źródłowy jądra, należy uruchomić `config(8)`.

```
# /usr/sbin/config MYKERNEL
```
2. Następnie, przenieśmy się do katalogu w którym dokonuje się budowy. Po ponownym uruchomieniu `config(8)` wyświetlona zostanie nazwa katalogu.

```
# cd ../compile/MYKERNEL
```

3. Skompilujmy jądro.

```
# make depend
# make
```

4. Zainstalujmy nowe jądro.

```
# make install
```

Procedura 2. Budowanie jądra w „nowy” sposób.

1. Wejdźmy do katalogu `/usr/src`.

```
# cd /usr/src
```

2. Skompilujmy jądro.

```
# make buildkernel KERNCONF=MYKERNEL
```

3. Zainstalujmy nowe jądro.

```
# make installkernel KERNCONF=MYKERNEL
```

Notatka: Ta metoda kompilacji jądra wymaga wszystkich plików źródłowych. Jeśli zainstalowaliśmy jedynie źródła jądra, powinniśmy skorzystać z opisaney powyżej metody tradycyjnej.

Podpowiedź: Domyślnie, podczas kompilacji indywidualnego jądra, *wszystkie* moduły jądra zostaną również zrekompileowane. Jeśli chcemy zaktualizować jądro szybciej bądź zbudować tylko własne moduły, powinniśmy przed rozpoczęciem kompilacji jądra zmodyfikować plik `/etc/make.conf`:

```
MODULES_OVERRIDE = linux acpi sound/sound sound/driver/dsl ntfs
```

Zmienna ta definiuje listę modułów do kompilacji zamiast wszystkich. Inne zmienne przydatne w procesie kompilacji jądra opisane zostały w podręczniku systemowym `make.conf(5)`.

Nowe jądro zostanie skopiowane do katalogu `/boot/kernel` jako `/boot/kernel/kernel`, a dotychczasowe zostanie przeniesione do `/boot/kernel.old/kernel`. Teraz należy ponownie uruchomić komputer. W razie jakby coś poszło źle, na końcu tego rozdziału przedstawionych zostało kilka awaryjnych rozwiązań. Przeczytajmy również rozdziały opisujące co zrobić w razie, gdy system nie chce się ponownie uruchomić.

Notatka: Inne pliki związane z procesem uruchamiania, np. takie jak `loader(8)` czy pliki konfiguracyjne są przechowywane w katalogu `/boot`. Własne moduły jak i moduły innych producentów, można umieszczać w katalogu `/boot/kernel`, jednakże użytkownicy powinni być świadomi, iż synchronizacja modułów ze skompilowanym jądrem jest bardzo ważna. Moduły nie przygotowane do pracy z danym jądrem mogą doprowadzić do niestabilności czy błędów.

8.4. Plik konfiguracyjny

Zaktualizował do FreeBSD 6.X Joel Dahl. Tłumaczył Mariusz Pilipczuk. Przekład uzupełnił Cezary Morga.

Ogólny format pliku konfiguracyjnego jest całkiem prosty. Każda linia zawiera słowo kluczowe i jeden lub więcej argumentów. Dla ułatwienia większość linii zawiera tylko jeden argument. Cokolwiek poprzedzone znakiem # jest uważane za komentarz i jest ignorowane. Ten rozdział opisuje każde słowo kluczowe w ogólnym porządku jaki zawiera plik `GENERIC`. Wyczerpująca lista opcji i więcej szczegółowych objaśnień zależnych od architektury znaleźć można w pliku `NOTES`, znajdującym się w tym samym katalogu co `GENERIC`. Opis opcji niezależnych od architektury znajduje się w pliku `/usr/src/sys/conf/NOTES`.

Notatka: By skompilować plik zawierający wszystkie dostępne opcje, jak się z reguły robi do celów testowych, należy wpisać jako `root` następujące polecenie:

```
# cd /usr/src/sys/i386/conf && make LINT
```

Poniżej opisany został przykład pliku konfiguracyjnego `GENERIC` z licznymi dodatkowymi komentarzami, tam gdzie są potrzebne objaśnienia. Przykład ten powinien odpowiadać naszej kopii pliku

```
/usr/src/sys/i386/conf/GENERIC.
```

```
machine          i386
```

Jest to architektura komputera. Musi być którymś z: `alpha`, `amd64`, `i386`, `ia64`, `pc98`, `powerpc`, lub `sparc64`.

```
cpu              I486_CPU
cpu              I586_CPU
cpu              I686_CPU
```

Powyższe wpisy określają typ CPU jaki posiadamy w swoim systemie. Możemy mieć kilka różnych wpisów (np. jeśli nie jesteśmy pewni czy mamy `I586_CPU` czy `I686_CPU`), jednak kiedy konfigurujemy jądro najlepiej pozostawić CPU jakie mamy. Jeśli nie jesteśmy pewni swojego procesora, możemy sprawdzić zawartość pliku `/var/run/dmesg.boot`, aby przejrzeć komunikaty startowe.

```
ident            GENERIC
```

Jest to identyfikator jądra. Możemy go zmienić na taki jak nazwaliśmy swoje jądro, w naszym poprzednim przykładzie `MYKERNEL`. Wartość jaką pozostawimy we wpisie `ident` będzie wyświetlana podczas startu, więc korzystnie jest dać nowemu jądro inną nazwę, jeśli chcemy go odróżnić od jądra, którego używamy na co dzień (np. chcemy zbudować eksperymentalne jądro).

```
#To statically compile in device wiring instead of /boot/device.hints
#hints            "GENERIC.hints"          # Default places to look for devices.
```

`device.hints(5)` jest wykorzystywany do konfiguracji opcji sterowników urządzeń. Domyślną lokacją sprawdzaną przez `loader(8)` w trakcie uruchamiania systemu jest `/boot/device.hints`. Wykorzystując opcję `hints` możemy wkompiłować je statycznie w jądro. Tym samym nie będzie potrzeby tworzyć pliku `device.hints` w katalogu `/boot`.

```
makeoptions      DEBUG=-g                  # Build kernel with gdb(1) debug symbols
```

Typowy proces kompilacji FreeBSD wyświetla również informacje diagnostyczne w trakcie budowy jądra z użyciem opcji `-g`, która włącza wyświetlanie informacji diagnostycznych w `gcc(1)`. Ten sam efekt można również osiągnąć poprzez opcję `-g` w `config(8)` przy korzystaniu z „tradycyjnej” metody kompilacji jądra (Sekcja 8.3 zawiera więcej informacji na temat budowy jądra).

```
options          SCHED_4BSD          # 4BSD scheduler
```

Tradycyjny i domyślny systemowy zarządca procesów FreeBSD. Nie zmieniamy tego.

```
options          PREEMPTION          # Enable kernel thread preemption
```

Pozwala na wyłaszczanie wątków w jądrze przez wątki o wyższym priorytecie. Pozwala to na interaktywność i przerywanie wątków, by ukończyć pewne czynności wcześniej i uniknąć oczekiwania.

```
options          INET                 # InterNETworking
```

Obsługa sieci. Należy pozostawić ten wpis, nawet jeśli nie planujemy podłączyć się do sieci. Większość programów wymaga przynajmniej urządzenia pętli zwrotnej loopback (np. tworzenie połączeń sieciowych wewnątrz naszego PC), więc jest to wpis bardzo istotny.

```
options          INET6                # IPv6 communications protocols
```

Umożliwia to obsługę protokołu komunikacyjnego IPv6.

```
options          FFS                  # Berkeley Fast Filesystem
```

Jest to podstawowy dyskowy system plików. Należy go pozostawić, jeśli startujemy system z dysku twardego.

```
options          SOFTUPDATES          # Enable FFS Soft Updates support
```

Opcja ta umożliwia tzw. Soft Updates w jądrze, co potrafi przyspieszyć czas dostępu do dysku przy zapisie. Jednakże, nawet jeśli funkcja ta jest włączona w jądrze, musi zostać aktywowana dla wybranych dysków. Czy opcja ta jest włączona możemy sprawdzić w wyniku polecenia `mount(8)`. Jeśli przy naszym dysku nie ma oznaczenia `soft-updates` oznacza to, że musimy ją włączyć wykorzystując polecenie `tunefs(8)` (dla istniejących systemów plików) bądź `newfs(8)` (dla nowych systemów plików).

```
options          UFS_ACL              # Support for access control lists
```

Opcja ta włącza w jądrze obsługę list kontroli dostępu do systemu plików. Polega to na wykorzystaniu rozszerzonych atrybutów oraz systemu plików UFS2. Sekcja 14.12 opisuje dokładniej tę funkcjonalność. Domyślnie listy ACL są włączone i nie powinny być wyłączone w jądrze jeśli były wcześniej wykorzystywane w systemie plików, gdyż usunie to listy kontroli dostępu zmieniając metodę ochrony plików w nieprzewidywalny sposób.

```
options          UFS_DIRHASH          # Improve performance on big directories
```

Opcja ta zawiera kod szybszej obsługi dużych katalogów kosztem zużycia dodatkowej pamięci. Możemy pozostawić tę opcję dla dużych serwerów lub dla interaktywnej stacji roboczej, a zablokować ją kiedy system jest mało obciążony i posiada mało pamięci, a dostęp do dysków nie jest taki ważny, np. serwer z zaporą ogniową.

```
options          MD_ROOT              # MD is a potential root device
```

Opcja ta włącza obsługę wirtualnego dysku w pamięci RAM, wykorzystywanego jako główne urządzenie.

```
options      NFSCLIENT      # Network Filesystem Client
options      NFSSERVER       # Network Filesystem Server
options      NFS_ROOT        # NFS usable as /, requires NFSCLIENT
```

Sieciowy system plików. Jeżeli nie planujemy montowania partycji z serwera UNIX poprzez TCP/IP, możemy zablokować te linie.

```
options      MSDOSFS         # MSDOS Filesystem
```

System plików MS-DOS. Jeśli nie planujemy montowania dysków lub partycji sformatowanych pod DOS-em podczas startowania systemu, dla bezpieczeństwa zablokujemy tę linię. Automatycznie MSDOSFS będzie ładowane kiedy pierwszy raz zamontujemy DOSową partycję jak opisano powyżej. Również wysmienity program `emulators/mtools` umożliwia dostęp do dyskietek DOSowych bez potrzeby ich montowania i odmontowywania (i bynajmniej nie jest potrzebny MSDOSFS).

```
options      CD9660          # ISO 9660 Filesystem
```

System plików ISO 9660 dla płyt CDROM. Jeśli nie posiadamy napędu CDROM możemy zablokować tę linię, lub gdy montujesz dane z CD okazjonalnie (od kiedy zamontujemy dane z CD po raz pierwszy, CD9660 będzie ładowany automatycznie). Płyty audio CD nie potrzebują tego systemu plików.

```
options      PROCFS          # Process filesystem (requires PSEUDofs)
```

System plików procesów. Jest to system plików „na niby” montowany w `/proc`, który dla takich programów jak `ps(1)` posiada więcej informacji o tym jakie procesy są właśnie uruchomione. W większości przypadków wykorzystanie PROCFS nie jest wymagane, gdyż większość narzędzi diagnostycznych i monitorujących zostało zaadaptowanych do pracy bez PROCFS: Domyślne instalacje nie montują tego systemu plików.

```
options      PSEUDofs        # Pseudo-filesystem framework
```

Jądra 6.X wykorzystujące PROCFS muszą również zawierać obsługę PSEUDofs.

```
options      GEOM_GPT        # GUID Partition Tables.
```

Opcja ta umożliwia tworzenie dużej ilości partycji na pojedynczym dysku.

```
options      COMPAT_43       # Compatible with BSD 4.3 [KEEP THIS!]
```

Kompatybilność z systemem 4.3BSD. Należy pozostawić ten wpis; niektóre programy będą zachowywać się dziwnie jeśli zablokujemy tę opcję.

```
options      COMPAT_FREEBSD4 # Compatible with FreeBSD4
```

Opcja ta potrzebna jest w systemach FreeBSD 5.X i386 i Alpha do obsługi aplikacji skompilowanych w starszych wersjach FreeBSD, wykorzystujących stary interfejs wywołań systemowych. Zaleca się by wykorzystywać tę opcję we wszystkich systemach i386 i Alpha, w których mogą wykorzystywane starsze aplikacje; platformy wspierane dopiero od wersji 5.X, jak np. ia64 i SPARC64®, nie wymagają ten opcji.

```
options      SCSI_DELAY=5000 # Delay (in ms) before probing SCSI
```

Sprawi to, że jądro zatrzyma się na 5 sekund przed rozpoczęciem rozpoznawania w naszym systemie każdego urządzenia SCSI. Jeśli jednak posiadamy tylko urządzenia IDE, możemy ten wpis zignorować. W innym przypadku

możemy zmniejszyć tę wartość i w ten sposób przyspieszyć start systemu. Gdy to zrobimy a FreeBSD będzie miał kłopoty z rozpoznawaniem urządzeń SCSI będziemy musieli zmienić tę wartość na większą.

```
options          KTRACE          # ktrace(1) support
```

Śledzenie procesów przez jądro, które jest użyteczne w diagnozowaniu.

```
options          SYSVSHM          # SYSV-style shared memory
```

Daje to systemom z rodziny V mechanizm współdzielenia pamięci. W działaniu ma to wiele wspólnego z mechanizmem XSHM w X-ach. Znaczna ilość programów obciążająca system graficzny zyska automatycznie na prędkości. Jeśli jesteśmy użytkownikiem X-ów koniecznie pozostawmy tę opcję.

```
options          SYSVMSG          # SYSV-style message queues
```

Wsparcie dla mechanizmu komunikatów w Systemach V. Opcja ta dodaje zaledwie kilkaset bajtów do jądra.

```
options          SYSVSEM          # SYSV-style semaphores
```

Wsparcie dla mechanizmu semaforów w Systemach V. Mniej przydatne w użyciu ale również dodaje tylko kilkaset bajtów do jądra.

Notatka: Parametr `-p` polecenia `ipcs(1)` wyświetli każdy proces, który używa tych dogodności Systemów V.

```
options          _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B real-time extensions
```

Rozszerzenia czasu rzeczywistego dodane w 1993 do POSIX®. Pewne aplikacje z kolekcji portów używają tego mechanizmu (jak np. **StarOffice**).

```
options          KBD_INSTALL_CDEV # install a CDEV entry in /dev
```

Opcja ta związana jest z obsługą klawiatury. Dodaje ona wpis CDEV w `/dev`.

```
options          AHC_REG_PRETTY_PRINT      # Print register bitfields in debug
                                                # output. Adds ~128k to driver.
```

```
options          AHD_REG_PRETTY_PRINT      # Print register bitfields in debug
                                                # output. Adds ~215k to driver.
```

Pomaga to w diagnozowaniu, wypisując łatwiejsze do odczytania definicje rejestrów.

```
options          ADAPTIVE_GIANT          # Giant mutex is adaptive.
```

Giant jest nazwą mechanizmu wzajemnego wykluczania (uśpiony mutex) chroniącego znaczną grupę zasobów jądra. Obecnie mechanizm ten stanowi niedopuszczalnie wąskie gardło w wydajności systemu, które jest zastępowane przez blokady zabezpieczające indywidualne zasoby. Opcja `ADAPTIVE_GIANT` powoduje, że Giant jest dołączany do zestawu adaptacyjnie zapętlanych muteksów. Co oznacza, że w momencie gdy wątek chce zablokować mutex Giant, który jest już zablokowany przez inny wątek bądź procesor, pierwszy wątek będzie pracował i oczekiwał na zwolnienie blokady. Normalnie, wątek przeszedłby do stanu uśpionia i oczekiwał na kolejną okazję uruchomienia. Jeśli nie jesteśmy przekonani, pozostawmy tę opcję włączoną.

```
device          apic              # I/O APIC
```

Urządzenie apic pozwala na wykorzystanie we/wy APIC do dostarczania przerw. Urządzenie apic może być wykorzystywane zarówno w jądrach UP jak i SMP, przy czym wymagane jest jedynie w przypadku tych drugich. By włączyć obsługę wielu procesorów należy dodać wiersz `options SMP`.

```
device          eisa
```

Należy włączyć to jeśli posiadamy płytę główną typu EISA. Umożliwia to autotestację i konfigurację dla wszystkich urządzeń pracujących na magistrali EISA.

```
device          pci
```

Włączmy to jeśli posiadamy płytę główną typu PCI. Umożliwia to autotestację kart PCI i przesyłanie z magistrali PCI do ISA.

```
# Floppy drives
device          fdc
```

Kontroler stacji dyskietek.

```
# ATA and ATAPI devices
device          ata
```

Sterownik ten obsługuje wszystkie urządzenia ATA i ATAPI. Potrzebujemy tylko tej jednej linijki, aby jądro wykrywało wszystkie urządzenia na współczesnych maszynach.

```
device          atadisk          # ATA disk drives
```

Potrzebne jest to razem z `device ata` dla dysków ATA.

```
device          ataraid          # ATA RAID drives
```

Potrzebne jest to razem z `device ata` dla dysków ATA RAID.

```
device          atapicd          # ATAPI CDROM drives
```

Potrzebne jest to razem z `device ata` dla napędów CDROM ATAPI.

```
device          atapifd          # ATAPI floppy drives
```

Potrzebne jest to razem z `device ata` dla stacji dyskietek ATAPI.

```
device          atapist          # ATAPI tape drives
```

Potrzebne jest to razem z `device ata` dla urządzeń taśmowych ATAPI.

```
options          ATA_STATIC_ID    # Static device numbering
```

Powoduje to przydzielanie przez kontroler statycznego numeru, inaczej liczba dyskowa będzie przydzielana dynamicznie.

```
# SCSI Controllers
device          ahb              # EISA AHA1742 family
device          ahc              # AHA2940 and onboard AIC7xxx devices
device          ahd              # AHA39320/29320 and onboard AIC79xx devices
```

```

device      amd      # AMD 53C974 (Teckram DC-390(T))
device      isp      # Qlogic family
#device     ispfw     # Firmware for QLogic HBAs- normally a module
device      mpt      # LSI-Logic MPT-Fusion
#device     ncr      # NCR/Symbios Logic
device      sym      # NCR/Symbios Logic (newer chipsets)
device      trm      # Tekram DC395U/UW/F DC315U adapters

device      adv      # Advansys SCSI adapters
device      adw      # Advansys wide SCSI adapters
device      aha      # Adaptec 154x SCSI adapters
device      aic      # Adaptec 15[012]x SCSI adapters, AIC-6[23]60.
device      bt       # Buslogic/Mylex MultiMaster SCSI adapters

device      ncv      # NCR 53C500
device      nsp      # Workbit Ninja SCSI-3
device      stg      # TMC 18C30/18C50

```

Kontrolery SCSI. Należy zablokować te kontrolery, których nie posiadamy w naszym systemie. Jeśli mamy system oparty tylko na IDE możemy pozbyć się całej listy.

```

# SCSI peripherals
device      scbus     # SCSI bus (required for SCSI)
device      ch        # SCSI media changers
device      da        # Direct Access (disks)
device      sa        # Sequential Access (tape etc)
device      cd        # CD
device      pass      # Passthrough device (direct SCSI access)
device      ses       # SCSI Environmental Services (and SAF-TE)

```

Peryferia SCSI. Ponownie, jeśli nie posiadamy takowych możemy je wyłączyć lub jeśli posiadamy tylko sprzęt IDE możemy wszystkie powyższe wpisy usunąć.

Notatka: Sterownik USB umass(4) i kilka innych sterowników wykorzystuje podsystem SCSI chociaż nie są one prawdziwymi urządzeniami SCSI. Tym samym musimy pamiętać by nie usunąć całkowicie obsługi SCSI jeśli którykolwiek z tego typu sterowników został uwzględniony w konfiguracji jądra.

```

# RAID controllers interfaced to the SCSI subsystem
device      amr       # AMI MegaRAID
device      arcmsr    # Areca SATA II RAID
device      asr       # DPT SmartRAID V, VI and Adaptec SCSI RAID
device      ciss      # Compaq Smart RAID 5*
device      dpt       # DPT Smartcache III, IV - See NOTES for options
device      hptmv     # Highpoint RocketRAID 182x
device      rr232x    # Highpoint RocketRAID 232x
device      iir       # Intel Integrated RAID
device      ips       # IBM (Adaptec) ServeRAID
device      mly       # Mylex AcceleRAID/eXtremeRAID
device      twa       # 3ware 9000 series PATA/SATA RAID

# RAID controllers

```

```

device      aac          # Adaptec FSA RAID
device      aacp         # SCSI passthrough for aac (requires CAM)
device      ida          # Compaq Smart RAID
device      mfi          # LSI MegaRAID SAS
device      mlx          # Mylex DAC960 family
device      pst          # Promise Supertrak SX6000
device      tve          # 3ware ATA RAID

```

Obsługa kontrolerów RAID. Jeśli nie posiadamy żadnych kontrolerów RAID, możemy te wpisy zablokować lub usunąć.

```

# atkbdc0 controls both the keyboard and the PS/2 mouse
device      atkbdc       # AT keyboard controller

```

Sterownik klawiatury (atkbdc) obsługujący porty we/wy dla klawiatur AT i dla urządzeń wskazujących PS/2. Wymagany jest przez sterownik klawiatur (atkbd) i PS/2 (psm).

```

device      atkbd        # AT keyboard

```

Sterownik atkbd razem z kontrolerem atkbdc umożliwiają dostęp do klawiatury AT84 lub do rozszerzonej klawiatury, które podłączone są do kontrolera AT.

```

device      psm          # PS/2 mouse

```

Urządzenie to należy wykorzystać jeśli nasza myszka jest podłączona do portu PS/2.

```

device      kbdmux       # keyboard multiplexer

```

Podstawowa obsługa multipleksacji klawiatury.

```

device      vga          # VGA video card driver

```

Sterownik kart video.

```

device      splash       # Splash screen and screen saver support

```

Obraz tytułowy w trakcie startu! Wymagany również przez wygaszacze ekranu.

```

# syscons is the default console driver, resembling an SCO console
device      sc

```

sc jest domyślnym sterownikiem konsoli, przypominający konsolę SCO. Wiele programów pracujących w trybie pełnoekranowym uzyskują dostęp do konsoli poprzez biblioteki bazy danych terminala takie jak termcap, nie powinno więc być istotne czy używamy właśnie jego czy vt, sterownika zgodnego z VT220. Kiedy logujemy się, a nasz program ma kłopoty podczas uruchamiania spod konsoli, należy ustawić zmienną TERM na scoansi.

```

# Enable this for the pcvt (VT220 compatible) console driver
#device      vt
#options     XSERVER      # support for X server on a vt console
#options     FAT_CURSOR   # start with block cursor

```

Sterowniki konsoli kompatybilnej z VT220 i z wcześniejszymi VT100/102. Dobrze pracują na niektórych laptopach nie posiadających sprzętu kompatybilnego z sc. Również w takim przypadku należy zmodyfikować zmienną TERM na vt100 lub vt220, kiedy się logujemy. Sterownik ten może być również użyteczny kiedy łączymy się z dużą

liczbą różnorodnych maszyn w sieci, gdzie `termcap` lub `terminfo` często nie posiadają wpisów dla urządzeń `sc` — wówczas `vt100` powinien być dostępny praktycznie na wszystkich platformach.

```
device          agp
```

Należy włączyć tę opcję jeśli posiadamy kartę AGP w systemie. Włączy to obsługę AGP i AGP GART dla płyt głównych obsługujących te funkcje.

```
# Power management support (see NOTES for more options)
#device          apm
```

Zaawansowane zarządzanie energią. Użyteczne dla laptopów, chociaż we FreeBSD 5.X i późniejszych opcja ta jest domyślnie wyłączona w jądrze `GENERIC`.

```
# Add suspend/resume support for the i8254.
device           pmtimer
```

Sterownik urządzenia regulatora czasowego dla zarządzania energią, jak np. APM i ACPI.

```
# PCCARD (PCMCIA) support
# PCMCIA and cardbus bridge support
device          cbb             # cardbus (yenta) bridge
device          pccard          # PC Card (16-bit) bus
device          cardbus         # CardBus (32-bit) bus
```

Obsługa kart PCMCIA. Potrzebna dla laptopów.

```
# Serial (COM) ports
device          sio             # 8250, 16[45]50 based serial ports
```

Są to porty szeregowo nazywane w terminologii MS-DOS/Windows `COM`.

Notatka: Jeśli posiadamy wewnętrzny modem na `COM4` oraz port szeregowy `COM2`, należy zmienić IRQ modemu na 2 (z technicznych pobudek `IRQ2 = IRQ9`) bo takiej kolejności wymaga FreeBSD. Jeśli posiadamy wieloportową kartę szeregową musimy odwołać się do podręcznika systemowego `sio(4)` po więcej informacji o właściwych ustawieniach w pliku `/boot/device.hints`. Niektóre karty wideo (zwłaszcza te bazujące na chipie S3) używają adresów `we/wy` w postaci `0x*2e8`, a ponieważ wiele tanich kart szeregowych nie dekoduje w pełni 16-bitowej przestrzeni adresowej `we/wy`, powodują one konflikt sprzętowy czyniąc port `COM4` praktycznie niedostępnym.

Każdy port szeregowy wymaga unikalnego IRQ (z wyjątkiem multiportów gdzie współdzielenie przerwania jest obsługiwane) zatem domyślne IRQ dla `COM3` i `COM4` nie mają zastosowania.

```
# Parallel port
device          ppc
```

Interfejs portu równoległego na magistrali ISA.

```
device          ppbus          # Parallel port bus (required)
```

Umożliwia obsługę portów równoległych.

```
device          lpt            # Printer
```

Obsługa drukarek na porcie równoległym.

Notatka: Powyższe trzy wpisy są wymagane, by było możliwe korzystanie z drukarek na porcie równoległym.

```
device      plip      # TCP/IP over parallel
```

Sterownik dla równoległego interfejsu sieciowego.

```
device      ppi      # Parallel port interface device
```

Uniwersalny port we/wy + IEEE1284.

```
#device     vpo      # Requires scbus and da
```

Napęd ZIP firmy Iomega. Wymagane sterowniki scbus i da. Najlepszą wydajność można osiągnąć wykorzystując porty w trybie EPP 1.9.

```
#device     puc
```

Opcję tę należy odblokować jeśli posiadamy „niemą” szeregową lub równoległą kartę PCI, obsługiwaną przez sterownik puc(4).

```
# PCI Ethernet NICs.
```

```
device      de        # DEC/Intel DC21x4x („Tulip”)
device      em        # Intel PRO/1000 adapter Gigabit Ethernet Card
device      ixgb      # Intel PRO/10GbE Ethernet Card
device      txp      # 3Com 3cR990 („Typhoon”)
device      vx        # 3Com 3c590, 3c595 („Vortex”)
```

Różne karty sieciowe na złączu PCI. Należy zablokować lub usunąć te z nich, które nie są obecne w naszym systemie.

```
# PCI Ethernet NICs that use the common MII bus controller code.
# NOTE: Be sure to keep the 'device miibus' line in order to use these NICs!
device      miibus    # MII bus support
```

Obsługa szyny MII wymagana dla wielu kart sieciowych 10/100 na złączu PCI, wykorzystujących nadajniki-odbiorniki zgodne z MII lub mają wbudowany nadbiornik pracujący jak MII. Dodanie device miibus do jądra pozwoli na obsługę miibus API i wszystkich sterowników PHY, włączając te, które nie wymagają indywidualnych ustawień i sterowników.

```
device      bce      # Broadcom BCM5706/BCM5708 Gigabit Ethernet
device      bfe      # Broadcom BCM440x 10/100 Ethernet
device      bge      # Broadcom BCM570xx Gigabit Ethernet
device      dc       # DEC/Intel 21143 and various workalikes
device      fxp      # Intel EtherExpress PRO/100B (82557, 82558)
device      lge      # Level 1 LXT1001 gigabit ethernet
device      nge      # NatSemi DP83820 gigabit ethernet
device      nve      # nVidia nForce MCP on-board Ethernet Networking
device      pcn      # AMD Am79C97x PCI 10/100 (precedence over 'lnc')
device      re       # RealTek 8139C+/8169/8169S/8110S
device      rl       # RealTek 8129/8139
```

```

device      sf          # Adaptec AIC-6915 („Starfire”)
device      sis         # Silicon Integrated Systems SiS 900/SiS 7016
device      sk          # SysKonnect SK-984x & SK-982x gigabit Ethernet
device      ste         # Sundance ST201 (D-Link DFE-550TX)
device      ti          # Alteon Networks Tigon I/II gigabit Ethernet
device      tl          # Texas Instruments ThunderLAN
device      tx          # SMC EtherPower II (83c170 „EPIC”)
device      vge         # VIA VT612x gigabit ethernet
device      vr          # VIA Rhine, Rhine II
device      wb          # Winbond W89C840F
device      xl          # 3Com 3c90x („Boomerang”, „Cyclone”)

```

Sterowniki wykorzystujące szynę MII.

```

# ISA Ethernet NICs. pccard NICs included.
device      cs          # Crystal Semiconductor CS89x0 NIC
# 'device ed' requires 'device miibus'
device      ed          # NE[12]000, SMC Ultra, 3c503, DS8390 cards
device      ex          # Intel EtherExpress Pro/10 and Pro/10+
device      ep          # Etherlink III based cards
device      fe          # Fujitsu MB8696x based cards
device      ie          # EtherExpress 8/16, 3C507, StarLAN 10 etc.
device      lnc         # NE2100, NE32-VL Lance Ethernet cards
device      sn          # SMC's 9000 series of Ethernet chips
device      xe          # Xircom pccard Ethernet

```

```

# ISA devices that use the old ISA shims
#device      le

```

Sterowniki ISA Ethernet. Plik `/usr/src/sys/i386/conf/NOTES` zawiera szczegółowy opis, która karta jest obsługiwana przez dany sterownik.

```

# Wireless NIC cards
device      wlan        # 802.11 support
device      an          # Aironet 4500/4800 802.11 wireless NICs.
device      awi         # BayStack 660 and others
device      ral         # Ralink Technology RT2500 wireless NICs.
device      wi          # WaveLAN/Intersil/Symbol 802.11 wireless NICs.
#device     wl          # Older non 802.11 Wavelan wireless NIC.

```

Obsługa różnych kart bezprzewodowych.

```

# Pseudo devices
device      loop        # Network loopback

```

Standardowe urządzenie pętli zwrotnej dla TCP/IP. Jeśli łączymy się z localhost (a.k.a. 127.0.0.1) za pomocą telnetu bądź FTP, połączenie powróci do nas za pomocą tego urządzenia. Obecność tego wpisu w konfiguracji jądra jest *niezbędna*.

```

device      random      # Entropy device

```

Bezpieczny z kryptograficznego punktu widzenia generator liczb losowych.

```

device      ether       # Ethernet support

```

ether jest wymagany tylko wówczas, gdy posiadamy kartę Ethernet. Zawiera podstawowy kod protokołu Ethernet.

```
device    sl                # Kernel SLIP
```

sl służy do obsługi SLIP. Zostało prawie całkowicie wyparte przez PPP, które jest łatwiejsze w obsłudze, lepiej przystosowane do połączeń modem - modem i posiada więcej możliwości.

```
device    ppp              # Kernel PPP
```

Wsparcie jądra dla PPP przy połączeniach wdzwanianych. Jest również w niej zaimplementowana wersja PPP, dla wielu aplikacji używających tun, oferująca większą elastyczność i funkcjonalności takie jak np. połączenie na żądanie (demand dialing).

```
device    tun              # Packet tunnel.
```

Używane przez rodzinę aplikacji korzystających z PPP. Więcej informacji na ten temat zawiera rozdział niniejszego Podręcznika poświęcony właśnie PPP.

```
device    pty              # Pseudo-ttys (telnet etc)
```

Jest to „pseudo-terminal” wykorzystywany przez przychodzące sesje telnet i rlogin, **xterm** oraz kilka innych aplikacji, jak np. **Emacs**.

```
device    md               # Memory „disks”
```

Pseudo urządzenie memory-disk.

```
device    gif              # IPv6 and IPv4 tunneling
```

Implementacja tunelowania IPv6 przez IPv4, IPv4 przez IPv6, IPv4 przez IPv4 oraz IPv6 przez IPv6. Urządzenie gif posiada cechę „auto-klonowania”, co umożliwia tworzenie wymaganych plików urządzeń.

```
device    faith            # IPv6-to-IPv4 relaying (translation)
```

To pseudo-urządzenie wyłapuje przesłane do niego pakiety i przekazuje je do demona translacji IPv4/IPv6.

```
# The 'bpf' device enables the Berkeley Packet Filter.
# Be aware of the administrative consequences of enabling this!
# Note that 'bpf' is required for DHCP.
device    bpf              # Berkeley packet filter
```

Filtr pakietów rodem z Berkeley. To pseudo-urządzenie pozwala interfejsom sieciowym pracować w trybie nasłuchiwania, wyłapując każdy pakiet wysłany w sieci (np w sieci Ethernet). Pakiety te mogą zostać zapisane na dysku i/lub sprawdzane programem tcpdump(1).

Notatka: Urządzenie bpf(4) jest również wykorzystywane przez dhclient(8), by uzyskać adres IP domyślnego rutera (bramki) itp. Jeśli używamy DHCP pozostawmy ten wpis.

```
# USB support
device    uhci              # UHCI PCI->USB interface
device    ohci              # OHCI PCI->USB interface
#device    ehci              # EHCI PCI->USB interface (USB 2.0)
```

```

device      usb          # USB Bus (required)
#device     udbp          # USB Double Bulk Pipe devices
device      ugen          # Generic
device      uhid          # „Human Interface Devices”
device      ukbd          # Keyboard
device      ulpt          # Printer
device      umass         # Disks/Mass storage - Requires scbus and da
device      ums           # Mouse
device      urio          # Diamond Rio 500 MP3 player
device      uscanner      # Scanners
# USB Ethernet, requires mii
device      aue           # ADMtek USB Ethernet
device      axe           # ASIX Electronics USB Ethernet
device      cdce          # Generic USB over Ethernet
device      cue           # CATC USB Ethernet
device      kue           # Kawasaki LSI USB Ethernet
device      rue           # RealTek RTL8150 USB Ethernet

```

Obsługa wielu urządzeń USB.

```

# FireWire support
device      firewire      # FireWire bus code
device      sbp           # SCSI over FireWire (Requires scbus and da)
device      fwe           # Ethernet over FireWire (non-standard!)

```

Obsługa różnorodnych urządzeń Firewire.

Więcej informacji o wymienionych oraz dodatkowych urządzeniach obsługiwanych przez FreeBSD znaleźć można w pliku `/usr/src/sys/i386/conf/NOTES`.

8.4.1. Konfiguracja dużego rozmiaru pamięci (PAE)

Maszyny dużego rozmiaru pamięci wymagają dostępu do większej ilości pamięci niż 4 gigabajty, do których ograniczona jest przestrzeń wirtualnych adresów użytkownika+jądra (ang. User+Kernel Virtual Address, KVA). Z tego właśnie powodu Intel dodał w procesorach serii Pentium Pro i późniejszych obsługę 36-bitowej przestrzeni adresów pamięci fizycznej.

Rozszerzenie PAE (ang. Physical Address Extension) procesorów Intel Pentium Pro i późniejszych pozwala na instalację do 64 gigabajtów pamięci. FreeBSD potrafi obsługiwać te rozszerzenie poprzez opcję konfiguracji jądra PAE, dostępną we wszystkich bieżących wersjach. Z uwagi na ograniczenia występujące w architekturze pamięci Intela, nie istnieje rozróżnienie pomiędzy rozmiarem pamięci poniżej i powyżej 4 gigabajtów. Pamięć znajdująca się powyżej jest po prostu dodawana do puli dostępnej pamięci.

By aktywować obsługę PAE w jądrze, wystarczy dodać poniższy wiersz do pliku konfiguracyjnego naszego jądra:

```
options      PAE
```

Notatka: Obsługa PAE jest dostępna we FreeBSD jedynie dla procesorów Intel IA-32. Należy również zwrócić uwagę, iż obsługa PAE we FreeBSD nie została szeroko przetestowana i powinna być traktowana jako drugiej jakości w porównaniu z innymi stabilnymi funkcjami FreeBSD.

Obsługa PAE we FreeBSD posiada również pewne ograniczenia:

- Dany proces nie ma dostępu do więcej jak 4 gigabajtów przestrzeni pamięci wirtualnej VM.
- Moduły KLD nie mogą być ładowane do jądra z włączoną opcją PAE, z uwagi na różnice w strukturze skompilowanego modułu i jądra.
- Sterowniki urządzeń nie wykorzystujące interfejsu `bus_dma(9)` spowodują utratę danych w jądrze z włączoną opcją PAE. Tym samym odradza się ich stosowanie. Z tego właśnie powodu plik konfiguracyjny jądra z opcją PAE jest dostarczany w wersji FreeBSD nie zawierającej żadnych ze sterowników, o których nie wiadomo, że współpracują poprawnie z jądrem z włączoną opcją PAE.
- Niektóre narzędzia dostrajania systemu określają wykorzystanie zasobów pamięci na podstawie ilości dostępnej pamięci fizycznej. Takie programy mogą niepotrzebnie przydzielać więcej pamięci niż powinny, z uwagi na naturę dużego rozmiaru pamięci systemu PAE. Przykładem może być opcja `sysctl kern.maxvnodes`, która kontroluje maksymalną liczbę dopuszczalnych węzłów w jądrze. Zaleca się modyfikację tych i innych parametrów do rozsądnych wartości.
- Może być potrzebnym zwiększenie rozmiaru przestrzeni adresów KVA bądź redukcja ilości specyficznych zasobów jądra często wykorzystywanych (patrz wyżej) w celu uniknięcia wyczerpania KVA. Do zwiększenia przestrzeni KVA może być wykorzystania opcja jądra `KVA_PAGES`.

W przypadku uwag odnośnie wydajności i stabilności pracy zaleca się lekturę podręcznika systemowego tuning(7). Podręcznik systemowy `pae(4)` zawiera natomiast aktualne informacje odnośnie obsługi PAE we FreeBSD.

8.5. Jeśli pojawią się kłopoty

Istnieje pięć kategorii problemów, które możemy napotkać budując jądro. Oto one:

Błąd config:

Jeśli program `config(8)` zgłosił błąd podczas przetwarzania naszego pliku konfiguracyjnego, najprawdopodobniej popełniliśmy mały błąd w postaci literówki. Na szczęście `config(8)` wyświetli linię, z którą miał problem, dzięki czemu będziemy mogli szybko do niej dotrzeć. Na przykład, jeśli widzimy:

```
config: line 17: syntax error
```

Upewnijmy się, że słowo kluczowe zostało poprawnie wprowadzone, porównując z oryginalnym plikiem `GENERIC` lub z innym wiarygodnym źródłem.

Błąd make:

Jeśli pojawił się błąd podczas wykonywania polecenia `make`, zwykle wskazuje to na błąd w naszym opisie jądra. Nie jest to jednak błąd na tyle wyraźny, aby wykazał go `config(8)`. Jak poprzednio, musimy przejrzeć plik konfiguracyjny jądra. Jeśli w dalszym ciągu nie możemy rozwiązać problemu, możemy wysłać nasz plik konfiguracyjny na Ogólną listę dyskusyjną FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) gdzie nasz problem zostanie rozwiązany bardzo szybko.

Jądro nie uruchamia się ponownie:

Jeśli nasze nowe jądro nie uruchamia się ponownie, bądź nie potrafi rozpoznać urządzeń, nie panikujmy! Na szczęście, FreeBSD jest wyposażone we wspaniały mechanizm przywracania po instalacji niekompatybilnego jądra. Po prostu musimy wybrać w loaderze jądro, które chcemy uruchomić. Możemy to zrobić, gdy system odlicza od 10 w dół. Wybieramy opcję numer sześć: „Escape to a loader prompt”. Wpisujemy `unload kernel` a następnie `boot /boot/kernel.old/kernel`, lub jakąkolwiek inną nazwę jądra, które uruchomi się poprawnie. Jeśli rekonfigurujemy jądro, jedno sprawne powinniśmy mieć zawsze pod ręką.

Po uruchomieniu z dobrym jądrem, możemy sprawdzić nasz plik konfiguracyjny, a następnie spróbować zbudować je ponownie. Pomocny jest plik `/var/log/messages`, w którym, pośród innych rzeczy, znajdują się również zapisy z uruchomień jądra. Ponadto również `dmesg(8)` wyświetla informacje z jądra, pochodzące z bieżącego uruchomienia.

Notatka: Jeśli mamy problemy ze zbudowaniem jądra, upewnijmy się, że posiadamy jądro `GENERIC` lub inne działające jądro nazwane tak, by nie zostało nadpisane po kolejnym procesie budowy. Nie możemy polegać na `kernel.old`, ponieważ gdy instalujemy nowe jądro, `kernel.old` jest nadpisywane przez ostatnio zainstalowane jądro, które może być niedziałające. Ponadto, powinniśmy tak szybko, jak to tylko możliwe, przenieść działające jądro do właściwej lokalizacji `/boot/kernel`, albo komendy takie jak `ps(1)` nie będą działały poprawnie. By to zrobić wystarczy zmienić nazwę katalogu zawierającego właściwe jądro:

```
# mv /boot/kernel /boot/kernel.bad
# mv /boot/kernel.good /boot/kernel
```

Jądro działa, ale przestało `ps(1)`:

Jeśli zainstalowaliśmy inną wersję jądra, niż tą, z którą były budowane narzędzia systemowe, na przykład jądro `-CURRENT` na systemie `-RELEASE`, wiele poleceń pokazujących stan systemu, jak `ps(1)`, czy `vmstat(8)` nie będzie działało. Musimy dokonać rekompilacji i instalacji `world` zbudowanych na podstawie tej samej wersji źródeł co nasze jądro. Jest to jeden z powodów, przez które nie jest najlepszym pomysłem instalowanie różnych wersji jądra i systemu operacyjnego.

Rozdział 9. Printing

Contributed by Sean Kelly. Restructured and updated by Jim Mock.

9.1. Synopsis

FreeBSD can be used to print with a wide variety of printers, from the oldest impact printer to the latest laser printers, and everything in between, allowing you to produce high-quality printed output from the applications you run.

FreeBSD can also be configured to act as a print server on a network; in this capacity FreeBSD can receive print jobs from a variety of other computers, including other FreeBSD computers, Windows and Mac OS hosts. FreeBSD will ensure that one job at a time is printed, and can keep statistics on which users and machines are doing the most printing, produce „banner” pages showing who’s printout is who’s, and more.

After reading this chapter, you will know:

- How to configure the FreeBSD print spooler.
- How to install print filters, to handle special print jobs differently, including converting incoming documents to print formats that your printers understand.
- How to enable header, or banner pages on your printout.
- How to print with printers connected to other computers.
- How to print with printers connected directly to the network.
- How to control printer restrictions, including limiting the size of print jobs, and preventing certain users from printing.
- How to keep printer statistics, and account for printer usage.
- How to troubleshoot printing problems.

Before reading this chapter, you should:

- Know how to configure and install a new kernel (Rozdział 8).

9.2. Introduction

In order to use printers with FreeBSD, you may set them up to work with the Berkeley line printer spooling system, also known as the **LPD** spooling system, or just **LPD**. It is the standard printer control system in FreeBSD. This chapter introduces **LPD** and will guide you through its configuration.

If you are already familiar with **LPD** or another printer spooling system, you may wish to skip to section Basic Setup.

LPD controls everything about a host’s printers. It is responsible for a number of things:

- It controls access to attached printers and printers attached to other hosts on the network.
- It enables users to submit files to be printed; these submissions are known as *jobs*.
- It prevents multiple users from accessing a printer at the same time by maintaining a *queue* for each printer.

- It can print *header pages* (also known as *banner* or *burst* pages) so users can easily find jobs they have printed in a stack of printouts.
- It takes care of communications parameters for printers connected on serial ports.
- It can send jobs over the network to a **LPD** spooler on another host.
- It can run special filters to format jobs to be printed for various printer languages or printer capabilities.
- It can account for printer usage.

Through a configuration file (`/etc/printcap`), and by providing the special filter programs, you can enable the **LPD** system to do all or some subset of the above for a great variety of printer hardware.

9.2.1. Why You Should Use the Spooler

If you are the sole user of your system, you may be wondering why you should bother with the spooler when you do not need access control, header pages, or printer accounting. While it is possible to enable direct access to a printer, you should use the spooler anyway since:

- **LPD** prints jobs in the background; you do not have to wait for data to be copied to the printer.
- **LPD** can conveniently run a job to be printed through filters to add date/time headers or convert a special file format (such as a \TeX DVI file) into a format the printer will understand. You will not have to do these steps manually.
- Many free and commercial programs that provide a print feature usually expect to talk to the spooler on your system. By setting up the spooling system, you will more easily support other software you may later add or already have.

9.3. Basic Setup

To use printers with the **LPD** spooling system, you will need to set up both your printer hardware and the **LPD** software. This document describes two levels of setup:

- See section Simple Printer Setup to learn how to connect a printer, tell **LPD** how to communicate with it, and print plain text files to the printer.
- See section Advanced Printer Setup to learn how to print a variety of special file formats, to print header pages, to print across a network, to control access to printers, and to do printer accounting.

9.3.1. Simple Printer Setup

This section tells how to configure printer hardware and the **LPD** software to use the printer. It teaches the basics:

- Section Hardware Setup gives some hints on connecting the printer to a port on your computer.
- Section Software Setup shows how to set up the **LPD** spooler configuration file (`/etc/printcap`).

If you are setting up a printer that uses a network protocol to accept data to print instead of a computer's local interfaces, see Printers With Networked Data Stream Interfaces.

Although this section is called „Simple Printer Setup”, it is actually fairly complex. Getting the printer to work with your computer and the **LPD** spooler is the hardest part. The advanced options like header pages and accounting are fairly easy once you get the printer working.

9.3.1.1. Hardware Setup

This section tells about the various ways you can connect a printer to your PC. It talks about the kinds of ports and cables, and also the kernel configuration you may need to enable FreeBSD to speak to the printer.

If you have already connected your printer and have successfully printed with it under another operating system, you can probably skip to section Software Setup.

9.3.1.1.1. Ports and Cables

Printers sold for use on PC's today generally come with one or more of the following three interfaces:

- *Serial* interfaces, also known as RS-232 or COM ports, use a serial port on your computer to send data to the printer. Serial interfaces are common in the computer industry and cables are readily available and also easy to construct. Serial interfaces sometimes need special cables and might require you to configure somewhat complex communications options. Most PC serial ports have a maximum transmission rate of 115200 bps, which makes printing large graphic print jobs with them impractical.
- *Parallel* interfaces use a parallel port on your computer to send data to the printer. Parallel interfaces are common in the PC market and are faster than RS-232 serial. Cables are readily available but more difficult to construct by hand. There are usually no communications options with parallel interfaces, making their configuration exceedingly simple.

Parallel interfaces are sometimes known as „Centronics” interfaces, named after the connector type on the printer.

- *USB* interfaces, named for the Universal Serial Bus, can run at even faster speeds than parallel or RS-232 serial interfaces. Cables are simple and cheap. USB is superior to RS-232 Serial and to Parallel for printing, but it is not as well supported under UNIX systems. A way to avoid this problem is to purchase a printer that has both a USB interface and a Parallel interface, as many printers do.

In general, Parallel interfaces usually offer just one-way communication (computer to printer) while serial and USB gives you two-way. Newer parallel ports (EPP and ECP) and printers can communicate in both directions under FreeBSD when a IEEE-1284-compliant cable is used.

Two-way communication to the printer over a parallel port is generally done in one of two ways. The first method uses a custom-built printer driver for FreeBSD that speaks the proprietary language used by the printer. This is common with inkjet printers and can be used for reporting ink levels and other status information. The second method is used when the printer supports PostScript.

PostScript jobs are actually programs sent to the printer; they need not produce paper at all and may return results directly to the computer. PostScript also uses two-way communication to tell the computer about problems, such as errors in the PostScript program or paper jams. Your users may be appreciative of such information. Furthermore, the best way to do effective accounting with a PostScript printer requires two-way communication: you ask the printer for its page count (how many pages it has printed in its lifetime), then send the user's job, then ask again for its page count. Subtract the two values and you know how much paper to charge to the user.

9.3.1.1.2. Parallel Ports

To hook up a printer using a parallel interface, connect the Centronics cable between the printer and the computer. The instructions that came with the printer, the computer, or both should give you complete guidance.

Remember which parallel port you used on the computer. The first parallel port is `ppc0` to FreeBSD; the second is `ppc1`, and so on. The printer device name uses the same scheme: `/dev/lpt0` for the printer on the first parallel ports etc.

9.3.1.1.3. Serial Ports

To hook up a printer using a serial interface, connect the proper serial cable between the printer and the computer. The instructions that came with the printer, the computer, or both should give you complete guidance.

If you are unsure what the „proper serial cable” is, you may wish to try one of the following alternatives:

- A *modem* cable connects each pin of the connector on one end of the cable straight through to its corresponding pin of the connector on the other end. This type of cable is also known as a „DTE-to-DCE” cable.
- A *null-modem* cable connects some pins straight through, swaps others (send data to receive data, for example), and shorts some internally in each connector hood. This type of cable is also known as a „DTE-to-DTE” cable.
- A *serial printer* cable, required for some unusual printers, is like the null-modem cable, but sends some signals to their counterparts instead of being internally shorted.

You should also set up the communications parameters for the printer, usually through front-panel controls or DIP switches on the printer. Choose the highest `bps` (bits per second, sometimes *baud rate*) that both your computer and the printer can support. Choose 7 or 8 data bits; none, even, or odd parity; and 1 or 2 stop bits. Also choose a flow control protocol: either none, or XON/XOFF (also known as „in-band” or „software”) flow control. Remember these settings for the software configuration that follows.

9.3.1.2. Software Setup

This section describes the software setup necessary to print with the **LPD** spooling system in FreeBSD.

Here is an outline of the steps involved:

1. Configure your kernel, if necessary, for the port you are using for the printer; section **Kernel Configuration** tells you what you need to do.
2. Set the communications mode for the parallel port, if you are using a parallel port; section **Setting the Communication Mode for the Parallel Port** gives details.
3. Test if the operating system can send data to the printer. Section **Checking Printer Communications** gives some suggestions on how to do this.
4. Set up **LPD** for the printer by modifying the file `/etc/printcap`. You will find out how to do this later in this chapter.

9.3.1.2.1. Kernel Configuration

The operating system kernel is compiled to work with a specific set of devices. The serial or parallel interface for your printer is a part of that set. Therefore, it might be necessary to add support for an additional serial or parallel port if your kernel is not already configured for one.

To find out if the kernel you are currently using supports a serial interface, type:

```
# grep sioN /var/run/dmesg.boot
```

Where N is the number of the serial port, starting from zero. If you see output similar to the following:

```
sio2 at port 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
```

then the kernel supports the port.

To find out if the kernel supports a parallel interface, type:

```
# grep ppcN /var/run/dmesg.boot
```

Where N is the number of the parallel port, starting from zero. If you see output similar to the following:

```
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/8 bytes threshold
```

then the kernel supports the port.

You might have to reconfigure your kernel in order for the operating system to recognize and use the parallel or serial port you are using for the printer.

To add support for a serial port, see the section on kernel configuration. To add support for a parallel port, see that section *and* the section that follows.

9.3.1.3. Setting the Communication Mode for the Parallel Port

When you are using the parallel interface, you can choose whether FreeBSD should use interrupt-driven or polled communication with the printer. The generic printer device driver (lpt(4)) on FreeBSD uses the pbus(4) system, which controls the port chipset with the ppc(4) driver.

- The *interrupt-driven* method is the default with the GENERIC kernel. With this method, the operating system uses an IRQ line to determine when the printer is ready for data.
- The *polled* method directs the operating system to repeatedly ask the printer if it is ready for more data. When it responds ready, the kernel sends more data.

The interrupt-driven method is usually somewhat faster but uses up a precious IRQ line. Some newer HP printers are claimed not to work correctly in interrupt mode, apparently due to some (not yet exactly understood) timing problem. These printers need polled mode. You should use whichever one works. Some printers will work in both modes, but are painfully slow in interrupt mode.

You can set the communications mode in two ways: by configuring the kernel or by using the lptcontrol(8) program.

To set the communications mode by configuring the kernel:

1. Edit your kernel configuration file. Look for an `ppc0` entry. If you are setting up the second parallel port, use `ppc1` instead. Use `ppc2` for the third port, and so on.
 - If you want interrupt-driven mode, edit the following line:


```
hint.ppc.0.irq="N"
```

 in the `/boot/device.hints` file and replace `N` with the right IRQ number. The kernel configuration file must also contain the `ppc(4)` driver:


```
device ppc
```
 - If you want polled mode, remove in your `/boot/device.hints` file, the following line:


```
hint.ppc.0.irq="N"
```

 In some cases, this is not enough to put the port in polled mode under FreeBSD. Most of time it comes from `acpi(4)` driver, this latter is able to probe and attach devices, and therefore, control the access mode to the printer port. You should check your `acpi(4)` configuration to correct this problem.
2. Save the file. Then configure, build, and install the kernel, then reboot. See kernel configuration for more details.

To set the communications mode with `lptcontrol(8)`:

1. Type:


```
# lptcontrol -i -d /dev/lptN
```

 to set interrupt-driven mode for `lptN`.
2. Type:


```
# lptcontrol -p -d /dev/lptN
```

 to set polled-mode for `lptN`.

You could put these commands in your `/etc/rc.local` file to set the mode each time your system boots. See `lptcontrol(8)` for more information.

9.3.1.4. Checking Printer Communications

Before proceeding to configure the spooling system, you should make sure the operating system can successfully send data to your printer. It is a lot easier to debug printer communication and the spooling system separately.

To test the printer, we will send some text to it. For printers that can immediately print characters sent to them, the program `lptest(1)` is perfect: it generates all 96 printable ASCII characters in 96 lines.

For a PostScript (or other language-based) printer, we will need a more sophisticated test. A small PostScript program, such as the following, will suffice:

```
%!PS
100 100 moveto 300 300 lineto stroke
310 310 moveto /Helvetica findfont 12 scalefont setfont
(Is this thing working?) show
showpage
```

The above PostScript code can be placed into a file and used as shown in the examples appearing in the following sections.

Notatka: When this document refers to a printer language, it is assuming a language like PostScript, and not Hewlett Packard's PCL. Although PCL has great functionality, you can intermingle plain text with its escape sequences. PostScript cannot directly print plain text, and that is the kind of printer language for which we must make special accommodations.

9.3.1.4.1. Checking a Parallel Printer

This section tells you how to check if FreeBSD can communicate with a printer connected to a parallel port.

To test a printer on a parallel port:

1. Become root with `su(1)`.

2. Send data to the printer.

- If the printer can print plain text, then use `lptest(1)`. Type:

```
# lptest > /dev/lptN
```

Where *N* is the number of the parallel port, starting from zero.

- If the printer understands PostScript or other printer language, then send a small program to the printer. Type:

```
# cat > /dev/lptN
```

Then, line by line, type the program *carefully* as you cannot edit a line once you have pressed RETURN or ENTER. When you have finished entering the program, press CONTROL+D, or whatever your end of file key is.

Alternatively, you can put the program in a file and type:

```
# cat file > /dev/lptN
```

Where *file* is the name of the file containing the program you want to send to the printer.

You should see something print. Do not worry if the text does not look right; we will fix such things later.

9.3.1.4.2. Checking a Serial Printer

This section tells you how to check if FreeBSD can communicate with a printer on a serial port.

To test a printer on a serial port:

1. Become root with `su(1)`.

2. Edit the file `/etc/remote`. Add the following entry:

```
printer:dv=/dev/port:br#bps-rate:pa=parity
```

Where *port* is the device entry for the serial port (`tttyd0`, `tttyd1`, etc.), *bps-rate* is the bits-per-second rate at which the printer communicates, and *parity* is the parity required by the printer (either even, odd, none, or zero).

Here is a sample entry for a printer connected via a serial line to the third serial port at 19200 bps with no parity:

```
printer:dv=/dev/ttyd2:br#19200:pa=none
```

3. Connect to the printer with `tip(1)`. Type:

```
# tip printer
```

If this step does not work, edit the file `/etc/remote` again and try using `/dev/cuaa0` instead of `/dev/ttyd0`.

4. Send data to the printer.

- If the printer can print plain text, then use `lpctest(1)`. Type:

```
% $lpctest
```

- If the printer understands PostScript or other printer language, then send a small program to the printer. Type the program, line by line, *very carefully* as backspacing or other editing keys may be significant to the printer. You may also need to type a special end-of-file key for the printer so it knows it received the whole program. For PostScript printers, press `CONTROL+D`.

Alternatively, you can put the program in a file and type:

```
% >file
```

Where *file* is the name of the file containing the program. After `tip(1)` sends the file, press any required end-of-file key.

You should see something print. Do not worry if the text does not look right; we will fix that later.

9.3.1.5. Enabling the Spooler: the `/etc/printcap` File

At this point, your printer should be hooked up, your kernel configured to communicate with it (if necessary), and you have been able to send some simple data to the printer. Now, we are ready to configure **LPD** to control access to your printer.

You configure **LPD** by editing the file `/etc/printcap`. The **LPD** spooling system reads this file each time the spooler is used, so updates to the file take immediate effect.

The format of the `printcap(5)` file is straightforward. Use your favorite text editor to make changes to `/etc/printcap`. The format is identical to other capability files like `/usr/share/misc/termcap` and `/etc/remote`. For complete information about the format, see the `cgetent(3)`.

The simple spooler configuration consists of the following steps:

1. Pick a name (and a few convenient aliases) for the printer, and put them in the `/etc/printcap` file; see the **Naming the Printer** section for more information on naming.
2. Turn off header pages (which are on by default) by inserting the `sh` capability; see the **Suppressing Header Pages** section for more information.
3. Make a spooling directory, and specify its location with the `sd` capability; see the **Making the Spooling Directory** section for more information.
4. Set the `/dev` entry to use for the printer, and note it in `/etc/printcap` with the `lp` capability; see the **Identifying the Printer Device** for more information. Also, if the printer is on a serial port, set up the communication parameters with the `ms#` capability which is discussed in the **Configuring Spooler Communications Parameters** section.
5. Install a plain text input filter; see the **Installing the Text Filter** section for details.

6. Test the setup by printing something with the `lpr(1)` command. More details are available in the Trying It Out and Troubleshooting sections.

Notatka: Language-based printers, such as PostScript printers, cannot directly print plain text. The simple setup outlined above and described in the following sections assumes that if you are installing such a printer you will print only files that the printer can understand.

Users often expect that they can print plain text to any of the printers installed on your system. Programs that interface to **LPD** to do their printing usually make the same assumption. If you are installing such a printer and want to be able to print jobs in the printer language *and* print plain text jobs, you are strongly urged to add an additional step to the simple setup outlined above: install an automatic plain-text-to-PostScript (or other printer language) conversion program. The section entitled Accommodating Plain Text Jobs on PostScript Printers tells how to do this.

9.3.1.5.1. Naming the Printer

The first (easy) step is to pick a name for your printer. It really does not matter whether you choose functional or whimsical names since you can also provide a number of aliases for the printer.

At least one of the printers specified in the `/etc/printcap` should have the alias `lp`. This is the default printer's name. If users do not have the `PRINTER` environment variable nor specify a printer name on the command line of any of the **LPD** commands, then `lp` will be the default printer they get to use.

Also, it is common practice to make the last alias for a printer be a full description of the printer, including make and model.

Once you have picked a name and some common aliases, put them in the `/etc/printcap` file. The name of the printer should start in the leftmost column. Separate each alias with a vertical bar and put a colon after the last alias.

In the following example, we start with a skeletal `/etc/printcap` that defines two printers (a Diablo 630 line printer and a Panasonic KX-P4455 PostScript laser printer):

```
#
# /etc/printcap for host rose
#
rattan|line|diablo|lp|Diablo 630 Line Printer:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:
```

In this example, the first printer is named `rattan` and has as aliases `line`, `diablo`, `lp`, and `Diablo 630 Line Printer`. Since it has the alias `lp`, it is also the default printer. The second is named `bamboo`, and has as aliases `ps`, `PS`, `S`, `panasonic`, and `Panasonic KX-P4455 PostScript v51.4`.

9.3.1.5.2. Suppressing Header Pages

The **LPD** spooling system will by default print a *header page* for each job. The header page contains the user name who requested the job, the host from which the job came, and the name of the job, in nice large letters. Unfortunately, all this extra text gets in the way of debugging the simple printer setup, so we will suppress header pages.

To suppress header pages, add the `sh` capability to the entry for the printer in `/etc/printcap`. Here is an example `/etc/printcap` with `sh` added:

```
#
# /etc/printcap for host rose - no header pages anywhere
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:
```

Note how we used the correct format: the first line starts in the leftmost column, and subsequent lines are indented. Every line in an entry except the last ends in a backslash character.

9.3.1.5.3. Making the Spooling Directory

The next step in the simple spooler setup is to make a *spooling directory*, a directory where print jobs reside until they are printed, and where a number of other spooler support files live.

Because of the variable nature of spooling directories, it is customary to put these directories under `/var/spool`. It is not necessary to backup the contents of spooling directories, either. Recreating them is as simple as running `mkdir(1)`.

It is also customary to make the directory with a name that is identical to the name of the printer, as shown below:

```
# mkdir /var/spool/printer-name
```

However, if you have a lot of printers on your network, you might want to put the spooling directories under a single directory that you reserve just for printing with **LPD**. We will do this for our two example printers `rattan` and `bamboo`:

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

Notatka: If you are concerned about the privacy of jobs that users print, you might want to protect the spooling directory so it is not publicly accessible. Spooling directories should be owned and be readable, writable, and searchable by user `daemon` and group `daemon`, and no one else. We will do this for our example printers:

```
# chown daemon:daemon /var/spool/lpd/rattan
# chown daemon:daemon /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan
# chmod 770 /var/spool/lpd/bamboo
```

Finally, you need to tell **LPD** about these directories using the `/etc/printcap` file. You specify the pathname of the spooling directory with the `sd` capability:

```
#
# /etc/printcap for host rose - added spooling directories
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:
```

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:
```

Note that the name of the printer starts in the first column but all other entries describing the printer should be indented and each line end escaped with a backslash.

If you do not specify a spooling directory with `sd`, the spooling system will use `/var/spool/lpd` as a default.

9.3.1.5.4. Identifying the Printer Device

In the Entries for the Ports section, we identified which entry in the `/dev` directory FreeBSD will use to communicate with the printer. Now, we tell **LPD** that information. When the spooling system has a job to print, it will open the specified device on behalf of the filter program (which is responsible for passing data to the printer).

List the `/dev` entry pathname in the `/etc/printcap` file using the `lp` capability.

In our running example, let us assume that `rattan` is on the first parallel port, and `bamboo` is on a sixth serial port; here are the additions to `/etc/printcap`:

```
#
# /etc/printcap for host rose - identified what devices to use
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
:sh:sd=/var/spool/lpd/rattan:\
:lp=/dev/lpt0:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:\
:lp=/dev/ttyd5:
```

If you do not specify the `lp` capability for a printer in your `/etc/printcap` file, **LPD** uses `/dev/lp` as a default. `/dev/lp` currently does not exist in FreeBSD.

If the printer you are installing is connected to a parallel port, skip to the section entitled, Installing the Text Filter. Otherwise, be sure to follow the instructions in the next section.

9.3.1.5.5. Configuring Spooler Communication Parameters

For printers on serial ports, **LPD** can set up the bps rate, parity, and other serial communication parameters on behalf of the filter program that sends data to the printer. This is advantageous since:

- It lets you try different communication parameters by simply editing the `/etc/printcap` file; you do not have to recompile the filter program.
- It enables the spooling system to use the same filter program for multiple printers which may have different serial communication settings.

The following `/etc/printcap` capabilities control serial communication parameters of the device listed in the `lp` capability:

`br#bps-rate`

Sets the communications speed of the device to *bps-rate*, where *bps-rate* can be 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, or 115200 bits-per-second.

`ms#stty-mode`

Sets the options for the terminal device after opening the device. `stty(1)` explains the available options.

When **LPD** opens the device specified by the `lp` capability, it sets the characteristics of the device to those specified with the `ms#` capability. Of particular interest will be the `parenb`, `parodd`, `cs5`, `cs6`, `cs7`, `cs8`, `cstopb`, `crtsets`, and `ixon` modes, which are explained in the `stty(1)` manual page.

Let us add to our example printer on the sixth serial port. We will set the bps rate to 38400. For the mode, we will set no parity with `-parenb`, 8-bit characters with `cs8`, no modem control with `cllocal` and hardware flow control with `crtsets`:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
      :sh:sd=/var/spool/lpd/bamboo:\
      :lp=/dev/ttyd5:ms#-parenb cs8 cllocal crtsets:
```

9.3.1.5.6. Installing the Text Filter

We are now ready to tell **LPD** what text filter to use to send jobs to the printer. A *text filter*, also known as an *input filter*, is a program that **LPD** runs when it has a job to print. When **LPD** runs the text filter for a printer, it sets the filter's standard input to the job to print, and its standard output to the printer device specified with the `lp` capability. The filter is expected to read the job from standard input, perform any necessary translation for the printer, and write the results to standard output, which will get printed. For more information on the text filter, see the Filters section.

For our simple printer setup, the text filter can be a small shell script that just executes `/bin/cat` to send the job to the printer. FreeBSD comes with another filter called `lpf` that handles backspacing and underlining for printers that might not deal with such character streams well. And, of course, you can use any other filter program you want. The filter `lpf` is described in detail in section entitled `lpf: a Text Filter`.

First, let us make the shell script `/usr/local/libexec/if-simple` be a simple text filter. Put the following text into that file with your favorite text editor:

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout. Ignores all filter arguments.

/bin/cat && exit 0
exit 2
```

Make the file executable:

```
# chmod 555 /usr/local/libexec/if-simple
```

And then tell **LPD** to use it by specifying it with the `if` capability in `/etc/printcap`. We will add it to the two printers we have so far in the example `/etc/printcap`:

```
#
# /etc/printcap for host rose - added text filter
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\ :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:\
    :if=/usr/local/libexec/if-simple:
```

Notatka: A copy of the `if-simple` script can be found in the `/usr/share/examples/printing` directory.

9.3.1.5.7. Turn on **LPD**

`lpd(8)` is run from `/etc/rc`, controlled by the `lpd_enable` variable. This variable defaults to `NO`. If you have not done so already, add the line:

```
lpd_enable="YES"
```

to `/etc/rc.conf`, and then either restart your machine, or just run `lpd(8)`.

```
# lpd
```

9.3.1.5.8. Trying It Out

You have reached the end of the simple **LPD** setup. Unfortunately, congratulations are not quite yet in order, since we still have to test the setup and correct any problems. To test the setup, try printing something. To print with the **LPD** system, you use the command `lpr(1)`, which submits a job for printing.

You can combine `lpr(1)` with the `lptest(1)` program, introduced in section Checking Printer Communications to generate some test text.

*To test the simple **LPD** setup:*

Type:

```
# lptest 20 5 | lpr -Pprinter-name
```

Where *printer-name* is the name of a printer (or an alias) specified in `/etc/printcap`. To test the default printer, type `lpr(1)` without any `-P` argument. Again, if you are testing a printer that expects PostScript, send a PostScript program in that language instead of using `lptest(1)`. You can do so by putting the program in a file and typing `lpr file`.

For a PostScript printer, you should get the results of the program. If you are using `lptest(1)`, then your results should look like the following:

```
! "#$%&' ( ) * + , - . / 0 1 2 3 4
" "#$%&' ( ) * + , - . / 0 1 2 3 4 5
```

```
#$%&'()*+,-./0123456
#$%&'()*+,-./01234567
%&'()*+,-./012345678
```

To further test the printer, try downloading larger programs (for language-based printers) or running `lpctest(1)` with different arguments. For example, `lpctest 80 60` will produce 60 lines of 80 characters each.

If the printer did not work, see the Troubleshooting section.

9.4. Advanced Printer Setup

This section describes filters for printing specially formatted files, header pages, printing across networks, and restricting and accounting for printer usage.

9.4.1. Filters

Although **LPD** handles network protocols, queuing, access control, and other aspects of printing, most of the *real* work happens in the *filters*. Filters are programs that communicate with the printer and handle its device dependencies and special requirements. In the simple printer setup, we installed a plain text filter—an extremely simple one that should work with most printers (section [Installing the Text Filter](#)).

However, in order to take advantage of format conversion, printer accounting, specific printer quirks, and so on, you should understand how filters work. It will ultimately be the filter’s responsibility to handle these aspects. And the bad news is that most of the time *you* have to provide filters yourself. The good news is that many are generally available; when they are not, they are usually easy to write.

Also, FreeBSD comes with one, `/usr/libexec/lpr/lpf`, that works with many printers that can print plain text. (It handles backspacing and tabs in the file, and does accounting, but that is about all it does.) There are also several filters and filter components in the FreeBSD Ports Collection.

Here is what you will find in this section:

- **Section How Filters Work**, tries to give an overview of a filter’s role in the printing process. You should read this section to get an understanding of what is happening „under the hood” when **LPD** uses filters. This knowledge could help you anticipate and debug problems you might encounter as you install more and more filters on each of your printers.
- **LPD** expects every printer to be able to print plain text by default. This presents a problem for PostScript (or other language-based printers) which cannot directly print plain text. Section [Accommodating Plain Text Jobs on PostScript Printers](#) tells you what you should do to overcome this problem. You should read this section if you have a PostScript printer.
- PostScript is a popular output format for many programs. Some people even write PostScript code directly. Unfortunately, PostScript printers are expensive. Section [Simulating PostScript on Non PostScript Printers](#) tells how you can further modify a printer’s text filter to accept and print PostScript data on a *non PostScript* printer. You should read this section if you do not have a PostScript printer.
- **Section Conversion Filters** tells about a way you can automate the conversion of specific file formats, such as graphic or typesetting data, into formats your printer can understand. After reading this section, you should be able

to set up your printers such that users can type `lpr -t` to print troff data, or `lpr -d` to print TeX DVI data, or `lpr -v` to print raster image data, and so forth. I recommend reading this section.

- Section **Output Filters** tells all about a not often used feature of **LPD**: output filters. Unless you are printing header pages (see **Header Pages**), you can probably skip that section altogether.
- Section **lpf: a Text Filter** describes `lpf`, a fairly complete if simple text filter for line printers (and laser printers that act like line printers) that comes with FreeBSD. If you need a quick way to get printer accounting working for plain text, or if you have a printer which emits smoke when it sees backspace characters, you should definitely consider `lpf`.

Notatka: A copy of the various scripts described below can be found in the `/usr/share/examples/printing` directory.

9.4.1.1. How Filters Work

As mentioned before, a filter is an executable program started by **LPD** to handle the device-dependent part of communicating with the printer.

When **LPD** wants to print a file in a job, it starts a filter program. It sets the filter's standard input to the file to print, its standard output to the printer, and its standard error to the error logging file (specified in the `lf` capability in `/etc/printcap`, or `/dev/console` by default).

Which filter **LPD** starts and the filter's arguments depend on what is listed in the `/etc/printcap` file and what arguments the user specified for the job on the `lpr(1)` command line. For example, if the user typed `lpr -t`, **LPD** would start the troff filter, listed in the `tf` capability for the destination printer. If the user wanted to print plain text, it would start the `if` filter (this is mostly true: see **Output Filters** for details).

There are three kinds of filters you can specify in `/etc/printcap`:

- The *text filter*, confusingly called the *input filter* in **LPD** documentation, handles regular text printing. Think of it as the default filter. **LPD** expects every printer to be able to print plain text by default, and it is the text filter's job to make sure backspaces, tabs, or other special characters do not confuse the printer. If you are in an environment where you have to account for printer usage, the text filter must also account for pages printed, usually by counting the number of lines printed and comparing that to the number of lines per page the printer supports. The text filter is started with the following argument list:

```
filter-name [-c] -wwidth -llength -iindent -n login -h host acct-file
```

where

`-c`

appears if the job is submitted with `lpr -l`

`width`

is the value from the `pw` (page width) capability specified in `/etc/printcap`, default 132

length

is the value from the `pl` (page length) capability, default 66

indent

is the amount of the indentation from `lpr -i`, default 0

login

is the account name of the user printing the file

host

is the host name from which the job was submitted

acct-file

is the name of the accounting file from the `af` capability.

- A *conversion filter* converts a specific file format into one the printer can render onto paper. For example, ditroff typesetting data cannot be directly printed, but you can install a conversion filter for ditroff files to convert the ditroff data into a form the printer can digest and print. Section Conversion Filters tells all about them. Conversion filters also need to do accounting, if you need printer accounting. Conversion filters are started with the following arguments:

```
filter-name -xpixel-width -ypixel-height -n login -h host acct-file
```

where *pixel-width* is the value from the `px` capability (default 0) and *pixel-height* is the value from the `py` capability (default 0).

- The *output filter* is used only if there is no text filter, or if header pages are enabled. In my experience, output filters are rarely used. Section Output Filters describe them. There are only two arguments to an output filter:

```
filter-name -wwidth -llength
```

which are identical to the text filters `-w` and `-l` arguments.

Filters should also *exit* with the following exit status:

exit 0

If the filter printed the file successfully.

exit 1

If the filter failed to print the file but wants **LPD** to try to print the file again. **LPD** will restart a filter if it exits with this status.

exit 2

If the filter failed to print the file and does not want **LPD** to try again. **LPD** will throw out the file.

The text filter that comes with the FreeBSD release, `/usr/libexec/lpr/lpf`, takes advantage of the page width and length arguments to determine when to send a form feed and how to account for printer usage. It uses the `login`, `host`, and accounting file arguments to make the accounting entries.

If you are shopping for filters, see if they are LPD-compatible. If they are, they must support the argument lists described above. If you plan on writing filters for general use, then have them support the same argument lists and exit codes.

9.4.1.2. Accommodating Plain Text Jobs on PostScript® Printers

If you are the only user of your computer and PostScript (or other language-based) printer, and you promise to never send plain text to your printer and to never use features of various programs that will want to send plain text to your printer, then you do not need to worry about this section at all.

But, if you would like to send both PostScript and plain text jobs to the printer, then you are urged to augment your printer setup. To do so, we have the text filter detect if the arriving job is plain text or PostScript. All PostScript jobs must start with `%!` (for other printer languages, see your printer documentation). If those are the first two characters in the job, we have PostScript, and can pass the rest of the job directly. If those are not the first two characters in the file, then the filter will convert the text into PostScript and print the result.

How do we do this?

If you have got a serial printer, a great way to do it is to install `lprps`. `lprps` is a PostScript printer filter which performs two-way communication with the printer. It updates the printer's status file with verbose information from the printer, so users and administrators can see exactly what the state of the printer is (such as `toner low` or `paper jam`). But more importantly, it includes a program called `psif` which detects whether the incoming job is plain text and calls `textps` (another program that comes with `lprps`) to convert it to PostScript. It then uses `lprps` to send the job to the printer.

`lprps` is part of the FreeBSD Ports Collection (see [The Ports Collection](#)). You can fetch, build and install it yourself, of course. After installing `lprps`, just specify the pathname to the `psif` program that is part of `lprps`. If you installed `lprps` from the Ports Collection, use the following in the serial PostScript printer's entry in `/etc/printcap`:

```
:if=/usr/local/libexec/psif:
```

You should also specify the `rw` capability; that tells **LPD** to open the printer in read-write mode.

If you have a parallel PostScript printer (and therefore cannot use two-way communication with the printer, which `lprps` needs), you can use the following shell script as the text filter:

```
#!/bin/sh
#
#  psif - Print PostScript or plain text on a PostScript printer
#  Script version; NOT the version that comes with lprps
#  Installed in /usr/local/libexec/psif
#

IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
#
#  PostScript job, print it.
#
echo "$first_line" && cat && printf "\004" && exit 0
exit 2
```

```

else
    #
    # Plain text, convert it, then print it.
    #
    ( echo "$first_line"; cat ) | /usr/local/bin/texttps && printf "\004" && exit 0
    exit 2
fi

```

In the above script, `texttps` is a program we installed separately to convert plain text to PostScript. You can use any text-to-PostScript program you wish. The FreeBSD Ports Collection (see [The Ports Collection](#)) includes a full featured text-to-PostScript program called `a2ps` that you might want to investigate.

9.4.1.3. Simulating PostScript on Non PostScript Printers

PostScript is the *de facto* standard for high quality typesetting and printing. PostScript is, however, an *expensive* standard. Thankfully, Aladdin Enterprises has a free PostScript work-alike called **Ghostscript** that runs with FreeBSD. Ghostscript can read most PostScript files and can render their pages onto a variety of devices, including many brands of non-PostScript printers. By installing Ghostscript and using a special text filter for your printer, you can make your non PostScript printer act like a real PostScript printer.

Ghostscript is in the FreeBSD Ports Collection, if you would like to install it from there. You can fetch, build, and install it quite easily yourself, as well.

To simulate PostScript, we have the text filter detect if it is printing a PostScript file. If it is not, then the filter will pass the file directly to the printer; otherwise, it will use Ghostscript to first convert the file into a format the printer will understand.

Here is an example: the following script is a text filter for Hewlett Packard DeskJet 500 printers. For other printers, substitute the `-sDEVICE` argument to the `gs` (Ghostscript) command. (Type `gs -h` to get a list of devices the current installation of Ghostscript supports.)

```

#!/bin/sh
#
# ifhp - Print Ghostscript-simulated PostScript on a DeskJet 500
# Installed in /usr/local/libexec/ifhp
#
# Treat LF as CR+LF (to avoid the "staircase effect" on HP/PCL
# printers):
#
printf "\033&k2G" || exit 2

#
# Read first two characters of the file
#
IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
    #
    # It is PostScript; use Ghostscript to scan-convert and print it.
    #
    /usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 \

```

```

        -sOutputFile=- - && exit 0
else
    #
    # Plain text or HP/PCL, so just print it directly; print a form feed
    # at the end to eject the last page.
    #
    echo "$first_line" && cat && printf "\033&l0H" &&
exit 0
fi

exit 2

```

Finally, you need to notify **LPD** of the filter via the `if` capability:

```
:if=/usr/local/libexec/ifhp:
```

That is it. You can type `lpr plain.text` and `lpr whatever.ps` and both should print successfully.

9.4.1.4. Conversion Filters

After completing the simple setup described in *Simple Printer Setup*, the first thing you will probably want to do is install conversion filters for your favorite file formats (besides plain ASCII text).

9.4.1.4.1. Why Install Conversion Filters?

Conversion filters make printing various kinds of files easy. As an example, suppose we do a lot of work with the \TeX typesetting system, and we have a PostScript printer. Every time we generate a DVI file from \TeX , we cannot print it directly until we convert the DVI file into PostScript. The command sequence goes like this:

```
% dvips seaweed-analysis.dvi
% lpr seaweed-analysis.ps
```

By installing a conversion filter for DVI files, we can skip the hand conversion step each time by having **LPD** do it for us. Now, each time we get a DVI file, we are just one step away from printing it:

```
% lpr -d seaweed-analysis.dvi
```

We got **LPD** to do the DVI file conversion for us by specifying the `-d` option. Section *Formatting and Conversion Options* lists the conversion options.

For each of the conversion options you want a printer to support, install a *conversion filter* and specify its pathname in `/etc/printcap`. A conversion filter is like the text filter for the simple printer setup (see section *Installing the Text Filter*) except that instead of printing plain text, the filter converts the file into a format the printer can understand.

9.4.1.4.2. Which Conversion Filters Should I Install?

You should install the conversion filters you expect to use. If you print a lot of DVI data, then a DVI conversion filter is in order. If you have got plenty of troff to print out, then you probably want a troff filter.

The following table summarizes the filters that **LPD** works with, their capability entries for the `/etc/printcap` file, and how to invoke them with the `lpr` command:

File type	/etc/printcap capability	lpr option
cifplot	cf	-c
DVI	df	-d
plot	gf	-g
ditroff	nf	-n
FORTTRAN text	rf	-f
troff	tf	-t
raster	vf	-v
plain text	if	none, -p, or -l

In our example, using `lpr -d` means the printer needs a `df` capability in its entry in `/etc/printcap`.

Despite what others might contend, formats like FORTRAN text and plot are probably obsolete. At your site, you can give new meanings to these or any of the formatting options just by installing custom filters. For example, suppose you would like to directly print Printerleaf files (files from the Interleaf desktop publishing program), but will never print plot files. You could install a Printerleaf conversion filter under the `gf` capability and then educate your users that `lpr -g` mean „print Printerleaf files.”

9.4.1.4.3. Installing Conversion Filters

Since conversion filters are programs you install outside of the base FreeBSD installation, they should probably go under `/usr/local`. The directory `/usr/local/libexec` is a popular location, since they are specialized programs that only **LPD** will run; regular users should not ever need to run them.

To enable a conversion filter, specify its pathname under the appropriate capability for the destination printer in `/etc/printcap`.

In our example, we will add the DVI conversion filter to the entry for the printer named `bamboo`. Here is the example `/etc/printcap` file again, with the new `df` capability for the printer `bamboo`.

```
#
# /etc/printcap for host rose - added df filter for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

The DVI filter is a shell script named `/usr/local/libexec/psdf`. Here is that script:

```
#!/bin/sh
#
# psdf - DVI to PostScript printer filter
# Installed in /usr/local/libexec/psdf
#
```

```
# Invoked by lpd when user runs lpr -d
#
exec /usr/local/bin/dvips -f | /usr/local/libexec/lprps "$@"
```

This script runs `dvips` in filter mode (the `-f` argument) on standard input, which is the job to print. It then starts the PostScript printer filter `lprps` (see section Accommodating Plain Text Jobs on PostScript Printers) with the arguments **LPD** passed to this script. `lprps` will use those arguments to account for the pages printed.

9.4.1.4.4. More Conversion Filter Examples

Since there is no fixed set of steps to install conversion filters, let me instead provide more examples. Use these as guidance to making your own filters. Use them directly, if appropriate.

This example script is a raster (well, GIF file, actually) conversion filter for a Hewlett Packard LaserJet III-Si printer:

```
#!/bin/sh
#
# hpvf - Convert GIF files into HP/PCL, then print
# Installed in /usr/local/libexec/hpvf

PATH=/usr/X11R6/bin:$PATH; export PATH
giftopnm | ppmtopgm | pgmtopbm | pbmtolj -resolution 300 \
    && exit 0 \
    || exit 2
```

It works by converting the GIF file into a portable anymap, converting that into a portable graymap, converting that into a portable bitmap, and converting that into LaserJet/PCL-compatible data.

Here is the `/etc/printcap` file with an entry for a printer using the above filter:

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:\
    :vf=/usr/local/libexec/hpvf:
```

The following script is a conversion filter for troff data from the groff typesetting system for the PostScript printer named bamboo:

```
#!/bin/sh
#
# pstf - Convert groff's troff data into PS, then print.
# Installed in /usr/local/libexec/pstf
#
exec grops | /usr/local/libexec/lprps "$@"
```

The above script makes use of `lprps` again to handle the communication with the printer. If the printer were on a parallel port, we would use this script instead:

```
#!/bin/sh
#
```

```
# pstf - Convert groff's troff data into PS, then print.
# Installed in /usr/local/libexec/pstf
#
exec grops
```

That is it. Here is the entry we need to add to `/etc/printcap` to enable the filter:

```
:tf=/usr/local/libexec/pstf:
```

Here is an example that might make old hands at FORTRAN blush. It is a FORTRAN-text filter for any printer that can directly print plain text. We will install it for the printer `teak`:

```
#!/bin/sh
#
# hprf - FORTRAN text filter for LaserJet 3si:
# Installed in /usr/local/libexec/hprf
#

printf "\033&k2G" && fpr && printf "\033&l0H" &&
    exit 0
exit 2
```

And we will add this line to the `/etc/printcap` for the printer `teak` to enable this filter:

```
:rf=/usr/local/libexec/hprf:
```

Here is one final, somewhat complex example. We will add a DVI filter to the LaserJet printer `teak` introduced earlier. First, the easy part: updating `/etc/printcap` with the location of the DVI filter:

```
:df=/usr/local/libexec/hpdf:
```

Now, for the hard part: making the filter. For that, we need a DVI-to-LaserJet/PCL conversion program. The FreeBSD Ports Collection (see [The Ports Collection](#)) has one: `dvi2xx` is the name of the package. Installing this package gives us the program we need, `dvilj2p`, which converts DVI into LaserJet IIp, LaserJet III, and LaserJet 2000 compatible codes.

`dvilj2p` makes the filter `hpdf` quite complex since `dvilj2p` cannot read from standard input. It wants to work with a filename. What is worse, the filename has to end in `.dvi` so using `/dev/fd/0` for standard input is problematic. We can get around that problem by linking (symbolically) a temporary file name (one that ends in `.dvi`) to `/dev/fd/0`, thereby forcing `dvilj2p` to read from standard input.

The only other fly in the ointment is the fact that we cannot use `/tmp` for the temporary link. Symbolic links are owned by user and group `bin`. The filter runs as user `daemon`. And the `/tmp` directory has the sticky bit set. The filter can create the link, but it will not be able clean up when done and remove it since the link will belong to a different user.

Instead, the filter will make the symbolic link in the current working directory, which is the spooling directory (specified by the `sd` capability in `/etc/printcap`). This is a perfect place for filters to do their work, especially since there is (sometimes) more free disk space in the spooling directory than under `/tmp`.

Here, finally, is the filter:

```
#!/bin/sh
#
```

```

# hpdf - Print DVI data on HP/PCL printer
# Installed in /usr/local/libexec/hpdf

PATH=/usr/local/bin:$PATH; export PATH

#
# Define a function to clean up our temporary files. These exist
# in the current directory, which will be the spooling directory
# for the printer.
#
cleanup() {
    rm -f hpdf$$dvi
}

#
# Define a function to handle fatal errors: print the given message
# and exit 2. Exiting with 2 tells LPD to do not try to reprint the
# job.
#
fatal() {
    echo "$@" 1>&2
    cleanup
    exit 2
}

#
# If user removes the job, LPD will send SIGINT, so trap SIGINT
# (and a few other signals) to clean up after ourselves.
#
trap cleanup 1 2 15

#
# Make sure we are not colliding with any existing files.
#
cleanup

#
# Link the DVI input file to standard input (the file to print).
#
ln -s /dev/fd/0 hpdf$$dvi || fatal "Cannot symlink /dev/fd/0"

#
# Make LF = CR+LF
#
printf "\033&k2G" || fatal "Cannot initialize printer"

#
# Convert and print. Return value from dvi2p does not seem to be
# reliable, so we ignore it.
#
dvi2p -M1 -q -e- dfhp$$dvi

#

```

```
# Clean up and exit
#
cleanup
exit 0
```

9.4.1.4.5. Automated Conversion: an Alternative to Conversion Filters

All these conversion filters accomplish a lot for your printing environment, but at the cost forcing the user to specify (on the `lpr(1)` command line) which one to use. If your users are not particularly computer literate, having to specify a filter option will become annoying. What is worse, though, is that an incorrectly specified filter option may run a filter on the wrong type of file and cause your printer to spew out hundreds of sheets of paper.

Rather than install conversion filters at all, you might want to try having the text filter (since it is the default filter) detect the type of file it has been asked to print and then automatically run the right conversion filter. Tools such as `file` can be of help here. Of course, it will be hard to determine the differences between *some* file types—and, of course, you can still provide conversion filters just for them.

The FreeBSD Ports Collection has a text filter that performs automatic conversion called `apsfilter`. It can detect plain text, PostScript, and DVI files, run the proper conversions, and print.

9.4.1.5. Output Filters

The **LPD** spooling system supports one other type of filter that we have not yet explored: an output filter. An output filter is intended for printing plain text only, like the text filter, but with many simplifications. If you are using an output filter but no text filter, then:

- **LPD** starts an output filter once for the entire job instead of once for each file in the job.
- **LPD** does not make any provision to identify the start or the end of files within the job for the output filter.
- **LPD** does not pass the user's login or host to the filter, so it is not intended to do accounting. In fact, it gets only two arguments:

```
filter-name -wwidth -llength
```

Where *width* is from the `pw` capability and *length* is from the `p1` capability for the printer in question.

Do not be seduced by an output filter's simplicity. If you would like each file in a job to start on a different page an output filter *will not work*. Use a text filter (also known as an input filter); see section [Installing the Text Filter](#). Furthermore, an output filter is actually *more complex* in that it has to examine the byte stream being sent to it for special flag characters and must send signals to itself on behalf of **LPD**.

However, an output filter is *necessary* if you want header pages and need to send escape sequences or other initialization strings to be able to print the header page. (But it is also *futile* if you want to charge header pages to the requesting user's account, since **LPD** does not give any user or host information to the output filter.)

On a single printer, **LPD** allows both an output filter and text or other filters. In such cases, **LPD** will start the output filter to print the header page (see section [Header Pages](#)) only. **LPD** then expects the output filter to *stop itself* by sending two bytes to the filter: ASCII 031 followed by ASCII 001. When an output filter sees these two bytes (031, 001), it should stop by sending `SIGSTOP` to itself. When **LPD**'s done running other filters, it will restart the output filter by sending `SIGCONT` to it.

If there is an output filter but *no* text filter and **LPD** is working on a plain text job, **LPD** uses the output filter to do the job. As stated before, the output filter will print each file of the job in sequence with no intervening form feeds or other paper advancement, and this is probably *not* what you want. In almost all cases, you need a text filter.

The program `lpf`, which we introduced earlier as a text filter, can also run as an output filter. If you need a quick-and-dirty output filter but do not want to write the byte detection and signal sending code, try `lpf`. You can also wrap `lpf` in a shell script to handle any initialization codes the printer might require.

9.4.1.6. `lpf`: a Text Filter

The program `/usr/libexec/lpr/lpf` that comes with FreeBSD binary distribution is a text filter (input filter) that can indent output (job submitted with `lpr -i`), allow literal characters to pass (job submitted with `lpr -l`), adjust the printing position for backspaces and tabs in the job, and account for pages printed. It can also act like an output filter.

`lpf` is suitable for many printing environments. And although it has no capability to send initialization sequences to a printer, it is easy to write a shell script to do the needed initialization and then execute `lpf`.

In order for `lpf` to do page accounting correctly, it needs correct values filled in for the `pw` and `pl` capabilities in the `/etc/printcap` file. It uses these values to determine how much text can fit on a page and how many pages were in a user's job. For more information on printer accounting, see *Accounting for Printer Usage*.

9.4.2. Header Pages

If you have *lots* of users, all of them using various printers, then you probably want to consider *header pages* as a necessary evil.

Header pages, also known as *banner* or *burst pages* identify to whom jobs belong after they are printed. They are usually printed in large, bold letters, perhaps with decorative borders, so that in a stack of printouts they stand out from the real documents that comprise users' jobs. They enable users to locate their jobs quickly. The obvious drawback to a header page is that it is yet one more sheet that has to be printed for every job, their ephemeral usefulness lasting not more than a few minutes, ultimately finding themselves in a recycling bin or rubbish heap. (Note that header pages go with each job, not each file in a job, so the paper waste might not be that bad.)

The **LPD** system can provide header pages automatically for your printouts *if* your printer can directly print plain text. If you have a PostScript printer, you will need an external program to generate the header page; see *Header Pages on PostScript Printers*.

9.4.2.1. Enabling Header Pages

In the *Simple Printer Setup* section, we turned off header pages by specifying `sh` (meaning „suppress header”) in the `/etc/printcap` file. To enable header pages for a printer, just remove the `sh` capability.

Sounds too easy, right?

You are right. You *might* have to provide an output filter to send initialization strings to the printer. Here is an example output filter for Hewlett Packard PCL-compatible printers:

```
#!/bin/sh
#
# hpof - Output filter for Hewlett Packard PCL-compatible printers
```

```
# Installed in /usr/local/libexec/hpof

printf "\033&k2G" || exit 2
exec /usr/libexec/lpr/lpf
```

Specify the path to the output filter in the `of` capability. See the Output Filters section for more information.

Here is an example `/etc/printcap` file for the printer `teak` that we introduced earlier; we enabled header pages and added the above output filter:

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:\
    :vf=/usr/local/libexec/hpvf:\
    :of=/usr/local/libexec/hpof:
```

Now, when users print jobs to `teak`, they get a header page with each job. If users want to spend time searching for their printouts, they can suppress header pages by submitting the job with `lpr -h`; see the Header Page Options section for more `lpr(1)` options.

Notatka: **LPD** prints a form feed character after the header page. If your printer uses a different character or sequence of characters to eject a page, specify them with the `ff` capability in `/etc/printcap`.

9.4.2.2. Controlling Header Pages

By enabling header pages, **LPD** will produce a *long header*, a full page of large letters identifying the user, host, and job. Here is an example (kelly printed the job named `outline` from host `rose`):

```

k          ll      ll
k          l       l
k          l       l
k  k      eeee     l       l      y   y
k  k      e   e    l       l      y   y
k k      eeeee    l       l      y   y
kk k      e        l       l      y   y
k  k      e   e    l       l      y  yy
k  k      eeee    ll      ll      yyy y
          Y
          Y   Y
          YYY
          YYY

          ll
          t       l       i
          t       l
o o o o  u   u  tttt  l       ii   n nnn  eeee
o   o   u   u   t    l       i     nn   n   e   e
```

```

o   o   u   u   t       l       i   n   n   eeeee
o   o   u   u   t       l       i   n   n   e
o   o   u   uu   t   t   l       i   n   n   e   e
oooo   uuu u       tt      lll     iii   n   n   eeee

```

```

r rrr      oooo      ssss      eeee
rr   r    o   o    s   s    e   e
r       o   o    ss       eeeeee
r       o   o      ss      e
r       o   o    s   s    e   e
r       oooo      ssss      eeee

```

```

Job:   outline
Date:  Sun Sep 17 11:04:58 1995

```

LPD appends a form feed after this text so the job starts on a new page (unless you have *sf* (suppress form feeds) in the destination printer's entry in */etc/printcap*).

If you prefer, **LPD** can make a *short header*; specify *sb* (short banner) in the */etc/printcap* file. The header page will look like this:

```

rose:kelly Job: outline Date: Sun Sep 17 11:07:51 1995

```

Also by default, **LPD** prints the header page first, then the job. To reverse that, specify *hl* (header last) in */etc/printcap*.

9.4.2.3. Accounting for Header Pages

Using **LPD**'s built-in header pages enforces a particular paradigm when it comes to printer accounting: header pages must be *free of charge*.

Why?

Because the output filter is the only external program that will have control when the header page is printed that could do accounting, and it is not provided with any *user or host* information or an accounting file, so it has no idea whom to charge for printer use. It is also not enough to just „add one page” to the text filter or any of the conversion filters (which do have user and host information) since users can suppress header pages with `lpr -h`. They could still be charged for header pages they did not print. Basically, `lpr -h` will be the preferred option of environmentally-minded users, but you cannot offer any incentive to use it.

It is *still not enough* to have each of the filters generate their own header pages (thereby being able to charge for them). If users wanted the option of suppressing the header pages with `lpr -h`, they will still get them and be charged for them since **LPD** does not pass any knowledge of the `-h` option to any of the filters.

So, what are your options?

You can:

- Accept **LPD**'s paradigm and make header pages free.
- Install an alternative to **LPD**, such as **LPRng**. Section Alternatives to the Standard Spooler tells more about other spooling software you can substitute for **LPD**.
- Write a *smart* output filter. Normally, an output filter is not meant to do anything more than initialize a printer or do some simple character conversion. It is suited for header pages and plain text jobs (when there is no text (input) filter). But, if there is a text filter for the plain text jobs, then **LPD** will start the output filter only for the header pages. And the output filter can parse the header page text that **LPD** generates to determine what user and host to charge for the header page. The only other problem with this method is that the output filter still does not know what accounting file to use (it is not passed the name of the file from the `af` capability), but if you have a well-known accounting file, you can hard-code that into the output filter. To facilitate the parsing step, use the `sh` (short header) capability in `/etc/printcap`. Then again, all that might be too much trouble, and users will certainly appreciate the more generous system administrator who makes header pages free.

9.4.2.4. Header Pages on PostScript Printers

As described above, **LPD** can generate a plain text header page suitable for many printers. Of course, PostScript cannot directly print plain text, so the header page feature of **LPD** is useless—or mostly so.

One obvious way to get header pages is to have every conversion filter and the text filter generate the header page. The filters should use the user and host arguments to generate a suitable header page. The drawback of this method is that users will always get a header page, even if they submit jobs with `lpr -h`.

Let us explore this method. The following script takes three arguments (user login name, host name, and job name) and makes a simple PostScript header page:

```
#!/bin/sh
#
# make-ps-header - make a PostScript header page on stdout
# Installed in /usr/local/libexec/make-ps-header
#
#
# These are PostScript units (72 to the inch).  Modify for A4 or
# whatever size paper you are using:
#
page_width=612
page_height=792
border=72
#
# Check arguments
#
if [ $# -ne 3 ]; then
```

```

    echo "Usage: `basename $0` <user> <host> <job>" 1>&2
    exit 1
fi

#
# Save these, mostly for readability in the PostScript, below.
#
user=$1
host=$2
job=$3
date=`date`

#
# Send the PostScript code to stdout.
#
exec cat <<EOF
%!PS

%
% Make sure we do not interfere with user's job that will follow
%
save

%
% Make a thick, unpleasant border around the edge of the paper.
%
$border $border moveto
$page_width $border 2 mul sub 0 rlineto
0 $page_height $border 2 mul sub rlineto
currentscreen 3 -1 roll pop 100 3 1 roll setscreen
$border 2 mul $page_width sub 0 rlineto closepath
0.8 setgray 10 setlinewidth stroke 0 setgray

%
% Display user's login name, nice and large and prominent
%
/Helvetica-Bold findfont 64 scalefont setfont
$page_width ($user) stringwidth pop sub 2 div $page_height 200 sub moveto
($user) show

%
% Now show the boring particulars
%
/Helvetica findfont 14 scalefont setfont
/y 200 def
[ (Job:) (Host:) (Date:) ] {
  200 y moveto show /y y 18 sub def }
forall

/Helvetica-Bold findfont 14 scalefont setfont
/y 200 def
[ ($job) ($host) ($date) ] {
  270 y moveto show /y y 18 sub def

```

```

} forall

%
% That is it
%
restore
showpage
EOF

```

Now, each of the conversion filters and the text filter can call this script to first generate the header page, and then print the user's job. Here is the DVI conversion filter from earlier in this document, modified to make a header page:

```

#!/bin/sh
#
# psdf - DVI to PostScript printer filter
# Installed in /usr/local/libexec/psdf
#
# Invoked by lpd when user runs lpr -d
#

orig_args="$@"

fail() {
    echo "$@" 1>&2
    exit 2
}

while getopts "x:y:n:h:" option; do
    case $option in
        x|y) ;; # Ignore
        n)   login=$OPTARG ;;
        h)   host=$OPTARG ;;
        *)   echo "LPD started `basename $0` wrong." 1>&2
            exit 2
            ;;
    esac
done

[ "$login" ] || fail "No login name"
[ "$host" ] || fail "No host name"

( /usr/local/libexec/make-ps-header $login $host "DVI File"
  /usr/local/bin/dvips -f ) | eval /usr/local/libexec/lprps $orig_args

```

Notice how the filter has to parse the argument list in order to determine the user and host name. The parsing for the other conversion filters is identical. The text filter takes a slightly different set of arguments, though (see section [How Filters Work](#)).

As we have mentioned before, the above scheme, though fairly simple, disables the „suppress header page” option (the `-h` option) to `lpr`. If users wanted to save a tree (or a few pennies, if you charge for header pages), they would not be able to do so, since every filter's going to print a header page with every job.

To allow users to shut off header pages on a per-job basis, you will need to use the trick introduced in section [Accounting for Header Pages](#): write an output filter that parses the LPD-generated header page and produces a PostScript version. If the user submits the job with `lpr -h`, then **LPD** will not generate a header page, and neither will your output filter. Otherwise, your output filter will read the text from **LPD** and send the appropriate header page PostScript code to the printer.

If you have a PostScript printer on a serial line, you can make use of `lprps`, which comes with an output filter, `psof`, which does the above. Note that `psof` does not charge for header pages.

9.4.3. Networked Printing

FreeBSD supports networked printing: sending jobs to remote printers. Networked printing generally refers to two different things:

- Accessing a printer attached to a remote host. You install a printer that has a conventional serial or parallel interface on one host. Then, you set up **LPD** to enable access to the printer from other hosts on the network. Section [Printers Installed on Remote Hosts](#) tells how to do this.
- Accessing a printer attached directly to a network. The printer has a network interface in addition (or in place of) a more conventional serial or parallel interface. Such a printer might work as follows:
 - It might understand the **LPD** protocol and can even queue jobs from remote hosts. In this case, it acts just like a regular host running **LPD**. Follow the same procedure in section [Printers Installed on Remote Hosts](#) to set up such a printer.
 - It might support a data stream network connection. In this case, you „attach” the printer to one host on the network by making that host responsible for spooling jobs and sending them to the printer. Section [Printers with Networked Data Stream Interfaces](#) gives some suggestions on installing such printers.

9.4.3.1. Printers Installed on Remote Hosts

The **LPD** spooling system has built-in support for sending jobs to other hosts also running **LPD** (or are compatible with **LPD**). This feature enables you to install a printer on one host and make it accessible from other hosts. It also works with printers that have network interfaces that understand the **LPD** protocol.

To enable this kind of remote printing, first install a printer on one host, the *printer host*, using the simple printer setup described in the [Simple Printer Setup](#) section. Do any advanced setup in [Advanced Printer Setup](#) that you need. Make sure to test the printer and see if it works with the features of **LPD** you have enabled. Also ensure that the *local host* has authorization to use the **LPD** service in the *remote host* (see [Restricting Jobs from Remote Printers](#)).

If you are using a printer with a network interface that is compatible with **LPD**, then the *printer host* in the discussion below is the printer itself, and the *printer name* is the name you configured for the printer. See the documentation that accompanied your printer and/or printer-network interface.

Podpowiedź: If you are using a Hewlett Packard Laserjet then the printer name `text` will automatically perform the LF to CRLF conversion for you, so you will not require the `hpif` script.

Then, on the other hosts you want to have access to the printer, make an entry in their `/etc/printcap` files with the following:

1. Name the entry anything you want. For simplicity, though, you probably want to use the same name and aliases as on the printer host.
2. Leave the `lp` capability blank, explicitly (`:lp=:`).
3. Make a spooling directory and specify its location in the `sd` capability. **LPD** will store jobs here before they get sent to the printer host.
4. Place the name of the printer host in the `rm` capability.
5. Place the printer name on the *printer host* in the `rp` capability.

That is it. You do not need to list conversion filters, page dimensions, or anything else in the `/etc/printcap` file.

Here is an example. The host `rose` has two printers, `bamboo` and `rattan`. We will enable users on the host `orchid` to print to those printers. Here is the `/etc/printcap` file for `orchid` (back from section **Enabling Header Pages**). It already had the entry for the printer `teak`; we have added entries for the two printers on the host `rose`:

```
#
# /etc/printcap for host orchid - added (remote) printers on rose
#

#
# teak is local; it is connected directly to orchid:
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/ifhp:\
    :vf=/usr/local/libexec/vfhp:\
    :of=/usr/local/libexec/ofhp:

#
# rattan is connected to rose; send jobs for rattan to rose:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

#
# bamboo is connected to rose as well:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:
```

Then, we just need to make spooling directories on `orchid`:

```
# mkdir -p /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chown daemon:daemon /var/spool/lpd/rattan /var/spool/lpd/bamboo
```

Now, users on `orchid` can print to `rattan` and `bamboo`. If, for example, a user on `orchid` typed

```
% lpr -P bamboo -d sushi-review.dvi
```

the **LPD** system on *orchid* would copy the job to the spooling directory */var/spool/lpd/bamboo* and note that it was a DVI job. As soon as the host *rose* has room in its *bamboo* spooling directory, the two **LPDs** would transfer the file to *rose*. The file would wait in *rose*'s queue until it was finally printed. It would be converted from DVI to PostScript (since *bamboo* is a PostScript printer) on *rose*.

9.4.3.2. Printers with Networked Data Stream Interfaces

Often, when you buy a network interface card for a printer, you can get two versions: one which emulates a spooler (the more expensive version), or one which just lets you send data to it as if you were using a serial or parallel port (the cheaper version). This section tells how to use the cheaper version. For the more expensive one, see the previous section *Printers Installed on Remote Hosts*.

The format of the */etc/printcap* file lets you specify what serial or parallel interface to use, and (if you are using a serial interface), what baud rate, whether to use flow control, delays for tabs, conversion of newlines, and more. But there is no way to specify a connection to a printer that is listening on a TCP/IP or other network port.

To send data to a networked printer, you need to develop a communications program that can be called by the text and conversion filters. Here is one such example: the script *netprint* takes all data on standard input and sends it to a network-attached printer. We specify the hostname of the printer as the first argument and the port number to which to connect as the second argument to *netprint*. Note that this supports one-way communication only (FreeBSD to printer); many network printers support two-way communication, and you might want to take advantage of that (to get printer status, perform accounting, etc.).

```
#!/usr/bin/perl
#
# netprint - Text filter for printer attached to network
# Installed in /usr/local/libexec/netprint
#
$#ARGV eq 1 || die "Usage: $0 <printer-hostname> <port-number>";

$printer_host = $ARGV[0];
$printer_port = $ARGV[1];

require 'sys/socket.ph';

($ignore, $ignore, $protocol) = getprotobyname('tcp');
($ignore, $ignore, $ignore, $ignore, $address)
    = gethostbyname($printer_host);

$sockaddr = pack('S n a4 x8', &AF_INET, $printer_port, $address);

socket(PRINTER, &PF_INET, &SOCK_STREAM, $protocol)
    || die "Can't create TCP/IP stream socket: $!";
connect(PRINTER, $sockaddr) || die "Can't contact $printer_host: $!";
while (<STDIN) { print PRINTER; }
exit 0;
```

We can then use this script in various filters. Suppose we had a Diablo 750-N line printer connected to the network. The printer accepts data to print on port number 5100. The host name of the printer is *scrivener*. Here is the text filter for the printer:

```
#!/bin/sh
```

```
#
# diablo-if-net - Text filter for Diablo printer 'scrivener' listening
# on port 5100. Installed in /usr/local/libexec/diablo-if-net
#
exec /usr/libexec/lpr/lpf "$@" | /usr/local/libexec/netprint scrivener 5100
```

9.4.4. Restricting Printer Usage

This section gives information on restricting printer usage. The **LPD** system lets you control who can access a printer, both locally or remotely, whether they can print multiple copies, how large their jobs can be, and how large the printer queues can get.

9.4.4.1. Restricting Multiple Copies

The **LPD** system makes it easy for users to print multiple copies of a file. Users can print jobs with `lpr -#5` (for example) and get five copies of each file in the job. Whether this is a good thing is up to you.

If you feel multiple copies cause unnecessary wear and tear on your printers, you can disable the `-#` option to `lpr(1)` by adding the `sc` capability to the `/etc/printcap` file. When users submit jobs with the `-#` option, they will see:

```
lpr: multiple copies are not allowed
```

Note that if you have set up access to a printer remotely (see section **Printers Installed on Remote Hosts**), you need the `sc` capability on the remote `/etc/printcap` files as well, or else users will still be able to submit multiple-copy jobs by using another host.

Here is an example. This is the `/etc/printcap` file for the host `rose`. The printer `rattan` is quite hearty, so we will allow multiple copies, but the laser printer `bamboo` is a bit more delicate, so we will disable multiple copies by adding the `sc` capability:

```
#
# /etc/printcap for host rose - restrict multiple copies on bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:\
    :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

Now, we also need to add the `sc` capability on the host `orchid`'s `/etc/printcap` (and while we are at it, let us disable multiple copies for the printer `teak`):

```
#
# /etc/printcap for host orchid - no multiple copies for local
# printer teak or remote printer bamboo
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
```

```

:lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:sc:\
:if=/usr/local/libexec/ifhp:\
:vf=/usr/local/libexec/vfhp:\
:of=/usr/local/libexec/ofhp:

rattan|line|diablo|lp|Diablo 630 Line Printer:\
:lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:sc:

```

By using the `sc` capability, we prevent the use of `lpr -#`, but that still does not prevent users from running `lpr(1)` multiple times, or from submitting the same file multiple times in one job like this:

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

There are many ways to prevent this abuse (including ignoring it) which you are free to explore.

9.4.4.2. Restricting Access to Printers

You can control who can print to what printers by using the UNIX group mechanism and the `rg` capability in `/etc/printcap`. Just place the users you want to have access to a printer in a certain group, and then name that group in the `rg` capability.

Users outside the group (including `root`) will be greeted with `lpr: Not a member of the restricted group` if they try to print to the controlled printer.

As with the `sc` (suppress multiple copies) capability, you need to specify `rg` on remote hosts that also have access to your printers, if you feel it is appropriate (see section [Printers Installed on Remote Hosts](#)).

For example, we will let anyone access the printer `rattan`, but only those in group `artists` can use `bamboo`. Here is the familiar `/etc/printcap` for host `rose`:

```

#
# /etc/printcap for host rose - restricted group for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
:sh:sd=/var/spool/lpd/rattan:\
:lp=/dev/lpt0:\
:if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:\
:lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
:if=/usr/local/libexec/psif:\
:df=/usr/local/libexec/psdf:

```

Let us leave the other example `/etc/printcap` file (for the host `orchid`) alone. Of course, anyone on `orchid` can print to `bamboo`. It might be the case that we only allow certain logins on `orchid` anyway, and want them to have access to the printer. Or not.

Notatka: There can be only one restricted group per printer.

9.4.4.3. Controlling Sizes of Jobs Submitted

If you have many users accessing the printers, you probably need to put an upper limit on the sizes of the files users can submit to print. After all, there is only so much free space on the filesystem that houses the spooling directories, and you also need to make sure there is room for the jobs of other users.

LPD enables you to limit the maximum byte size a file in a job can be with the `mx` capability. The units are in `BUFSIZ` blocks, which are 1024 bytes. If you put a zero for this capability, there will be no limit on file size; however, if no `mx` capability is specified, then a default limit of 1000 blocks will be used.

Notatka: The limit applies to *files* in a job, and *not* the total job size.

LPD will not refuse a file that is larger than the limit you place on a printer. Instead, it will queue as much of the file up to the limit, which will then get printed. The rest will be discarded. Whether this is correct behavior is up for debate.

Let us add limits to our example printers `rattan` and `bamboo`. Since those artists' PostScript files tend to be large, we will limit them to five megabytes. We will put no limit on the plain text line printer:

```
#
# /etc/printcap for host rose
#

#
# No limit on job size:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:mx#0:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

#
# Limit of five megabytes:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
    :lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

Again, the limits apply to the local users only. If you have set up access to your printers remotely, remote users will not get those limits. You will need to specify the `mx` capability in the remote `/etc/printcap` files as well. See section **Printers Installed on Remote Hosts** for more information on remote printing.

There is another specialized way to limit job sizes from remote printers; see section **Restricting Jobs from Remote Printers**.

9.4.4.4. Restricting Jobs from Remote Printers

The **LPD** spooling system provides several ways to restrict print jobs submitted from remote hosts:

Host restrictions

You can control from which remote hosts a local **LPD** accepts requests with the files `/etc/hosts.equiv` and `/etc/hosts.lpd`. **LPD** checks to see if an incoming request is from a host listed in either one of these files. If not, **LPD** refuses the request.

The format of these files is simple: one host name per line. Note that the file `/etc/hosts.equiv` is also used by the `ruserok(3)` protocol, and affects programs like `rsh(1)` and `rcp(1)`, so be careful.

For example, here is the `/etc/hosts.lpd` file on the host `rose`:

```
orchid
violet
madrigal.fishbaum.de
```

This means `rose` will accept requests from the hosts `orchid`, `violet`, and `madrigal.fishbaum.de`. If any other host tries to access `rose`'s **LPD**, the job will be refused.

Size restrictions

You can control how much free space there needs to remain on the filesystem where a spooling directory resides. Make a file called `minfree` in the spooling directory for the local printer. Insert in that file a number representing how many disk blocks (512 bytes) of free space there has to be for a remote job to be accepted.

This lets you insure that remote users will not fill your filesystem. You can also use it to give a certain priority to local users: they will be able to queue jobs long after the free disk space has fallen below the amount specified in the `minfree` file.

For example, let us add a `minfree` file for the printer `bamboo`. We examine `/etc/printcap` to find the spooling directory for this printer; here is `bamboo`'s entry:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
:lp=/dev/ttyd5:ms#-parenb cs8 clocal crtscts:rw:mx#5000:\
:if=/usr/local/libexec/psif:\
:df=/usr/local/libexec/psdf:
```

The spooling directory is given in the `sd` capability. We will make three megabytes (which is 6144 disk blocks) the amount of free disk space that must exist on the filesystem for **LPD** to accept remote jobs:

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

User restrictions

You can control which remote users can print to local printers by specifying the `rs` capability in `/etc/printcap`. When `rs` appears in the entry for a locally-attached printer, **LPD** will accept jobs from remote hosts *if* the user submitting the job also has an account of the same login name on the local host. Otherwise, **LPD** refuses the job.

This capability is particularly useful in an environment where there are (for example) different departments sharing a network, and some users transcend departmental boundaries. By giving them accounts on your systems, they can use your printers from their own departmental systems. If you would rather allow them to use

only your printers and not your computer resources, you can give them „token” accounts, with no home directory and a useless shell like `/usr/bin/false`.

9.4.5. Accounting for Printer Usage

So, you need to charge for printouts. And why not? Paper and ink cost money. And then there are maintenance costs—printers are loaded with moving parts and tend to break down. You have examined your printers, usage patterns, and maintenance fees and have come up with a per-page (or per-foot, per-meter, or per-whatever) cost. Now, how do you actually start accounting for printouts?

Well, the bad news is the **LPD** spooling system does not provide much help in this department. Accounting is highly dependent on the kind of printer in use, the formats being printed, and *your* requirements in charging for printer usage.

To implement accounting, you have to modify a printer’s text filter (to charge for plain text jobs) and the conversion filters (to charge for other file formats), to count pages or query the printer for pages printed. You cannot get away with using the simple output filter, since it cannot do accounting. See section Filters.

Generally, there are two ways to do accounting:

- *Periodic accounting* is the more common way, possibly because it is easier. Whenever someone prints a job, the filter logs the user, host, and number of pages to an accounting file. Every month, semester, year, or whatever time period you prefer, you collect the accounting files for the various printers, tally up the pages printed by users, and charge for usage. Then you truncate all the logging files, starting with a clean slate for the next period.
- *Timely accounting* is less common, probably because it is more difficult. This method has the filters charge users for printouts as soon as they use the printers. Like disk quotas, the accounting is immediate. You can prevent users from printing when their account goes in the red, and might provide a way for users to check and adjust their „print quotas.” But this method requires some database code to track users and their quotas.

The **LPD** spooling system supports both methods easily: since you have to provide the filters (well, most of the time), you also have to provide the accounting code. But there is a bright side: you have enormous flexibility in your accounting methods. For example, you choose whether to use periodic or timely accounting. You choose what information to log: user names, host names, job types, pages printed, square footage of paper used, how long the job took to print, and so forth. And you do so by modifying the filters to save this information.

9.4.5.1. Quick and Dirty Printer Accounting

FreeBSD comes with two programs that can get you set up with simple periodic accounting right away. They are the text filter `lpf`, described in section `lpf: a Text Filter`, and `pac(8)`, a program to gather and total entries from printer accounting files.

As mentioned in the section on filters (Filters), **LPD** starts the text and the conversion filters with the name of the accounting file to use on the filter command line. The filters can use this argument to know where to write an accounting file entry. The name of this file comes from the `af` capability in `/etc/printcap`, and if not specified as an absolute path, is relative to the spooling directory.

LPD starts `lpf` with page width and length arguments (from the `pw` and `pl` capabilities). `lpf` uses these arguments to determine how much paper will be used. After sending the file to the printer, it then writes an accounting entry in the accounting file. The entries look like this:

```

2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang

```

You should use a separate accounting file for each printer, as `lpd` has no file locking logic built into it, and two `lpds` might corrupt each other's entries if they were to write to the same file at the same time. An easy way to insure a separate accounting file for each printer is to use `af=acct` in `/etc/printcap`. Then, each accounting file will be in the spooling directory for a printer, in a file named `acct`.

When you are ready to charge users for printouts, run the `pac(8)` program. Just change to the spooling directory for the printer you want to collect on and type `pac`. You will get a dollar-centric summary like the following:

Login	pages/feet	runs	price
orchid:kelly	5.00	1	\$ 0.10
orchid:mary	31.00	3	\$ 0.62
orchid:zhang	9.00	1	\$ 0.18
rose:andy	2.00	1	\$ 0.04
rose:kelly	177.00	104	\$ 3.54
rose:mary	87.00	32	\$ 1.74
rose:root	26.00	12	\$ 0.52
total	337.00	154	\$ 6.74

These are the arguments `pac(8)` expects:

`-Pprinter`

Which *printer* to summarize. This option works only if there is an absolute path in the `af` capability in `/etc/printcap`.

`-c`

Sort the output by cost instead of alphabetically by user name.

`-m`

Ignore host name in the accounting files. With this option, user `smith` on host `alpha` is the same user `smith` on host `gamma`. Without, they are different users.

`-pprice`

Compute charges with *price* dollars per page or per foot instead of the price from the `pc` capability in `/etc/printcap`, or two cents (the default). You can specify *price* as a floating point number.

`-r`

Reverse the sort order.

`-s`

Make an accounting summary file and truncate the accounting file.

name . . .

Print accounting information for the given user *names* only.

In the default summary that `pac(8)` produces, you see the number of pages printed by each user from various hosts. If, at your site, host does not matter (because users can use any host), run `pac -m`, to produce the following summary:

Login	pages/foot	runs	price
andy	2.00	1	\$ 0.04
kelly	182.00	105	\$ 3.64
mary	118.00	35	\$ 2.36
root	26.00	12	\$ 0.52
zhang	9.00	1	\$ 0.18
total	337.00	154	\$ 6.74

To compute the dollar amount due, `pac(8)` uses the `pc` capability in the `/etc/printcap` file (default of 200, or 2 cents per page). Specify, in hundredths of cents, the price per page or per foot you want to charge for printouts in this capability. You can override this value when you run `pac(8)` with the `-p` option. The units for the `-p` option are in dollars, though, not hundredths of cents. For example,

```
# pac -p1.50
```

makes each page cost one dollar and fifty cents. You can really rake in the profits by using this option.

Finally, running `pac -s` will save the summary information in a summary accounting file, which is named the same as the printer's accounting file, but with `_sum` appended to the name. It then truncates the accounting file. When you run `pac(8)` again, it rereads the summary file to get starting totals, then adds information from the regular accounting file.

9.4.5.2. How Can You Count Pages Printed?

In order to perform even remotely accurate accounting, you need to be able to determine how much paper a job uses. This is the essential problem of printer accounting.

For plain text jobs, the problem is not that hard to solve: you count how many lines are in a job and compare it to how many lines per page your printer supports. Do not forget to take into account backspaces in the file which overprint lines, or long logical lines that wrap onto one or more additional physical lines.

The text filter `lpf` (introduced in `lpf: a Text Filter`) takes into account these things when it does accounting. If you are writing a text filter which needs to do accounting, you might want to examine `lpf`'s source code.

How do you handle other file formats, though?

Well, for DVI-to-LaserJet or DVI-to-PostScript conversion, you can have your filter parse the diagnostic output of `dvi1j` or `dvips` and look to see how many pages were converted. You might be able to do similar things with other file formats and conversion programs.

But these methods suffer from the fact that the printer may not actually print all those pages. For example, it could jam, run out of toner, or explode—and the user would still get charged.

So, what can you do?

There is only one *sure* way to do *accurate* accounting. Get a printer that can tell you how much paper it uses, and attach it via a serial line or a network connection. Nearly all PostScript printers support this notion. Other makes and

models do as well (networked Imagen laser printers, for example). Modify the filters for these printers to get the page usage after they print each job and have them log accounting information based on that value *only*. There is no line counting nor error-prone file examination required.

Of course, you can always be generous and make all printouts free.

9.5. Using Printers

This section tells you how to use printers you have set up with FreeBSD. Here is an overview of the user-level commands:

`lpr(1)`

Print jobs

`lpq(1)`

Check printer queues

`lprm(1)`

Remove jobs from a printer's queue

There is also an administrative command, `lpc(8)`, described in the section *Administering Printers*, used to control printers and their queues.

All three of the commands `lpr(1)`, `lprm(1)`, and `lpq(1)` accept an option `-P printer-name` to specify on which printer/queue to operate, as listed in the `/etc/printcap` file. This enables you to submit, remove, and check on jobs for various printers. If you do not use the `-P` option, then these commands use the printer specified in the `PRINTER` environment variable. Finally, if you do not have a `PRINTER` environment variable, these commands default to the printer named `lp`.

Hereafter, the terminology *default printer* means the printer named in the `PRINTER` environment variable, or the printer named `lp` when there is no `PRINTER` environment variable.

9.5.1. Printing Jobs

To print files, type:

```
% lpr filename ...
```

This prints each of the listed files to the default printer. If you list no files, `lpr(1)` reads data to print from standard input. For example, this command prints some important system files:

```
% lpr /etc/host.conf /etc/hosts.equiv
```

To select a specific printer, type:

```
% lpr -P printer-name filename ...
```

This example prints a long listing of the current directory to the printer named `rattan`:

```
% ls -l | lpr -P rattan
```

Because no files were listed for the `lpr(1)` command, `lpr` read the data to print from standard input, which was the output of the `ls -l` command.

The `lpr(1)` command can also accept a wide variety of options to control formatting, apply file conversions, generate multiple copies, and so forth. For more information, see the section [Printing Options](#).

9.5.2. Checking Jobs

When you print with `lpr(1)`, the data you wish to print is put together in a package called a „print job”, which is sent to the **LPD** spooling system. Each printer has a queue of jobs, and your job waits in that queue along with other jobs from yourself and from other users. The printer prints those jobs in a first-come, first-served order.

To display the queue for the default printer, type `lpq(1)`. For a specific printer, use the `-P` option. For example, the command

```
% lpq -P bamboo
```

shows the queue for the printer named `bamboo`. Here is an example of the output of the `lpq` command:

```
bamboo is ready and printing
Rank  Owner   Job  Files                                Total Size
active kelly   9    /etc/host.conf, /etc/hosts.equiv    88 bytes
2nd    kelly   10    (standard input)                   1635 bytes
3rd    mary    11    ...                                78519 bytes
```

This shows three jobs in the queue for `bamboo`. The first job, submitted by user `kelly`, got assigned „job number” 9. Every job for a printer gets a unique job number. Most of the time you can ignore the job number, but you will need it if you want to cancel the job; see section [Removing Jobs](#) for details.

Job number nine consists of two files; multiple files given on the `lpr(1)` command line are treated as part of a single job. It is the currently active job (note the word `active` under the „Rank” column), which means the printer should be currently printing that job. The second job consists of data passed as the standard input to the `lpr(1)` command. The third job came from user `mary`; it is a much larger job. The pathname of the file she is trying to print is too long to fit, so the `lpq(1)` command just shows three dots.

The very first line of the output from `lpq(1)` is also useful: it tells what the printer is currently doing (or at least what **LPD** thinks the printer is doing).

The `lpq(1)` command also support a `-l` option to generate a detailed long listing. Here is an example of `lpq -l`:

```
waiting for bamboo to become ready (offline ?)
kelly: 1st                                [job 009rose]
      /etc/host.conf                      73 bytes
      /etc/hosts.equiv                   15 bytes

kelly: 2nd                                [job 010rose]
      (standard input)                   1635 bytes

mary: 3rd                                [job 011rose]
      /home/orchid/mary/research/venus/alpha-regio/mapping 78519 bytes
```

9.5.3. Removing Jobs

If you change your mind about printing a job, you can remove the job from the queue with the `lprm(1)` command. Often, you can even use `lprm(1)` to remove an active job, but some or all of the job might still get printed.

To remove a job from the default printer, first use `lpq(1)` to find the job number. Then type:

```
% lprm job-number
```

To remove the job from a specific printer, add the `-P` option. The following command removes job number 10 from the queue for the printer `bamboo`:

```
% lprm -P bamboo 10
```

The `lprm(1)` command has a few shortcuts:

`lprm -`

Removes all jobs (for the default printer) belonging to you.

`lprm user`

Removes all jobs (for the default printer) belonging to *user*. The superuser can remove other users' jobs; you can remove only your own jobs.

`lprm`

With no job number, user name, or `-` appearing on the command line, `lprm(1)` removes the currently active job on the default printer, if it belongs to you. The superuser can remove any active job.

Just use the `-P` option with the above shortcuts to operate on a specific printer instead of the default. For example, the following command removes all jobs for the current user in the queue for the printer named `rattan`:

```
% lprm -P rattan -
```

Notatka: If you are working in a networked environment, `lprm(1)` will let you remove jobs only from the host from which the jobs were submitted, even if the same printer is available from other hosts. The following command sequence demonstrates this:

```
% lpr -P rattan myfile
% rlogin orchid
% lpq -P rattan
Rank  Owner      Job  Files                Total Size
active seeyan    12   ...                49123 bytes
2nd   kelly       13   myfile              12 bytes
% lprm -P rattan 13
rose: Permission denied
% logout
% lprm -P rattan 13
dfA013rose dequeued
cfA013rose dequeued
```

9.5.4. Beyond Plain Text: Printing Options

The `lpr(1)` command supports a number of options that control formatting text, converting graphic and other file formats, producing multiple copies, handling of the job, and more. This section describes the options.

9.5.4.1. Formatting and Conversion Options

The following `lpr(1)` options control formatting of the files in the job. Use these options if the job does not contain plain text or if you want plain text formatted through the `pr(1)` utility.

For example, the following command prints a DVI file (from the \TeX typesetting system) named `fish-report.dvi` to the printer named `bamboo`:

```
% lpr -P bamboo -d fish-report.dvi
```

These options apply to every file in the job, so you cannot mix (say) DVI and ditroff files together in a job. Instead, submit the files as separate jobs, using a different conversion option for each job.

Notatka: All of these options except `-p` and `-t` require conversion filters installed for the destination printer. For example, the `-d` option requires the DVI conversion filter. Section [Conversion Filters](#) gives details.

`-c`

Print cifplot files.

`-d`

Print DVI files.

`-f`

Print FORTRAN text files.

`-g`

Print plot data.

`-i number`

Indent the output by *number* columns; if you omit *number*, indent by 8 columns. This option works only with certain conversion filters.

Notatka: Do not put any space between the `-i` and the number.

`-l`

Print literal text data, including control characters.

-n

Print ditroff (device independent troff) data.

-P

Format plain text with `pr(1)` before printing. See `pr(1)` for more information.

-T *title*

Use *title* on the `pr(1)` header instead of the file name. This option has effect only when used with the `-p` option.

-t

Print troff data.

-v

Print raster data.

Here is an example: this command prints a nicely formatted version of the `ls(1)` manual page on the default printer:

```
% zcat /usr/share/man/man1/ls.1.gz | troff -t -man | lpr -t
```

The `zcat(1)` command uncompresses the source of the `ls(1)` manual page and passes it to the `troff(1)` command, which formats that source and makes GNU troff output and passes it to `lpr(1)`, which submits the job to the **LPD** spooler. Because we used the `-t` option to `lpr(1)`, the spooler will convert the GNU troff output into a format the default printer can understand when it prints the job.

9.5.4.2. Job Handling Options

The following options to `lpr(1)` tell **LPD** to handle the job specially:

-# *copies*

Produce a number of *copies* of each file in the job instead of just one copy. An administrator may disable this option to reduce printer wear-and-tear and encourage photocopier usage. See section [Restricting Multiple Copies](#).

This example prints three copies of `parser.c` followed by three copies of `parser.h` to the default printer:

```
% lpr -#3 parser.c parser.h
```

-m

Send mail after completing the print job. With this option, the **LPD** system will send mail to your account when it finishes handling your job. In its message, it will tell you if the job completed successfully or if there was an error, and (often) what the error was.

-s

Do not copy the files to the spooling directory, but make symbolic links to them instead.

If you are printing a large job, you probably want to use this option. It saves space in the spooling directory (your job might overflow the free space on the filesystem where the spooling directory resides). It saves time as well since **LPD** will not have to copy each and every byte of your job to the spooling directory.

There is a drawback, though: since **LPD** will refer to the original files directly, you cannot modify or remove them until they have been printed.

Notatka: If you are printing to a remote printer, **LPD** will eventually have to copy files from the local host to the remote host, so the `-s` option will save space only on the local spooling directory, not the remote. It is still useful, though.

`-r`

Remove the files in the job after copying them to the spooling directory, or after printing them with the `-s` option. Be careful with this option!

9.5.4.3. Header Page Options

These options to `lpr(1)` adjust the text that normally appears on a job's header page. If header pages are suppressed for the destination printer, these options have no effect. See section Header Pages for information about setting up header pages.

`-C text`

Replace the hostname on the header page with *text*. The hostname is normally the name of the host from which the job was submitted.

`-J text`

Replace the job name on the header page with *text*. The job name is normally the name of the first file of the job, or `stdin` if you are printing standard input.

`-h`

Do not print any header page.

Notatka: At some sites, this option may have no effect due to the way header pages are generated. See Header Pages for details.

9.5.5. Administering Printers

As an administrator for your printers, you have had to install, set up, and test them. Using the `lpc(8)` command, you can interact with your printers in yet more ways. With `lpc(8)`, you can

- Start and stop the printers
- Enable and disable their queues

- Rearrange the order of the jobs in each queue.

First, a note about terminology: if a printer is *stopped*, it will not print anything in its queue. Users can still submit jobs, which will wait in the queue until the printer is *started* or the queue is cleared.

If a queue is *disabled*, no user (except `root`) can submit jobs for the printer. An *enabled* queue allows jobs to be submitted. A printer can be *started* for a disabled queue, in which case it will continue to print jobs in the queue until the queue is empty.

In general, you have to have `root` privileges to use the `lpc(8)` command. Ordinary users can use the `lpc(8)` command to get printer status and to restart a hung printer only.

Here is a summary of the `lpc(8)` commands. Most of the commands take a *printer-name* argument to tell on which printer to operate. You can use `all` for the *printer-name* to mean all printers listed in `/etc/printcap`.

`abort printer-name`

Cancel the current job and stop the printer. Users can still submit jobs if the queue is enabled.

`clean printer-name`

Remove old files from the printer's spooling directory. Occasionally, the files that make up a job are not properly removed by **LPD**, particularly if there have been errors during printing or a lot of administrative activity. This command finds files that do not belong in the spooling directory and removes them.

`disable printer-name`

Disable queuing of new jobs. If the printer is running, it will continue to print any jobs remaining in the queue. The superuser (`root`) can always submit jobs, even to a disabled queue.

This command is useful while you are testing a new printer or filter installation: disable the queue and submit jobs as `root`. Other users will not be able to submit jobs until you complete your testing and re-enable the queue with the `enable` command.

`down printer-name message`

Take a printer down. Equivalent to `disable` followed by `stop`. The *message* appears as the printer's status whenever a user checks the printer's queue with `lpq(1)` or status with `lpc status`.

`enable printer-name`

Enable the queue for a printer. Users can submit jobs but the printer will not print anything until it is started.

`help command-name`

Print help on the command *command-name*. With no *command-name*, print a summary of the commands available.

`restart printer-name`

Start the printer. Ordinary users can use this command if some extraordinary circumstance hangs **LPD**, but they cannot start a printer stopped with either the `stop` or `down` commands. The `restart` command is equivalent to `abort` followed by `start`.

`start printer-name`

Start the printer. The printer will print jobs in its queue.

```
stop printer-name
```

Stop the printer. The printer will finish the current job and will not print anything else in its queue. Even though the printer is stopped, users can still submit jobs to an enabled queue.

```
topq printer-name job-or-username
```

Rearrange the queue for *printer-name* by placing the jobs with the listed *job* numbers or the jobs belonging to *username* at the top of the queue. For this command, you cannot use *all* as the *printer-name*.

```
up printer-name
```

Bring a printer up; the opposite of the *down* command. Equivalent to *start* followed by *enable*.

lpc(8) accepts the above commands on the command line. If you do not enter any commands, *lpc(8)* enters an interactive mode, where you can enter commands until you type *exit*, *quit*, or end-of-file.

9.6. Alternatives to the Standard Spooler

If you have been reading straight through this manual, by now you have learned just about everything there is to know about the **LPD** spooling system that comes with FreeBSD. You can probably appreciate many of its shortcomings, which naturally leads to the question: „What other spooling systems are out there (and work with FreeBSD)?”

LPRng

LPRng, which purportedly means „LPR: the Next Generation” is a complete rewrite of PLP. Patrick Powell and Justin Mason (the principal maintainer of PLP) collaborated to make **LPRng**. The main site for **LPRng** is <http://www.lprng.org/>.

CUPS

CUPS, the Common UNIX Printing System, provides a portable printing layer for UNIX-based operating systems. It has been developed by Easy Software Products to promote a standard printing solution for all UNIX vendors and users.

CUPS uses the Internet Printing Protocol (IPP) as the basis for managing print jobs and queues. The Line Printer Daemon (LPD), Server Message Block (SMB), and AppSocket (a.k.a. JetDirect) protocols are also supported with reduced functionality. CUPS adds network printer browsing and PostScript Printer Description (PPD) based printing options to support real-world printing under UNIX.

The main site for **CUPS** is <http://www.cups.org/>.

9.7. Troubleshooting

After performing the simple test with *lptest(1)*, you might have gotten one of the following results instead of the correct printout:

It worked, after awhile; or, it did not eject a full sheet.

The printer printed the above, but it sat for awhile and did nothing. In fact, you might have needed to press a PRINT REMAINING or FORM FEED button on the printer to get any results to appear.

If this is the case, the printer was probably waiting to see if there was any more data for your job before it printed anything. To fix this problem, you can have the text filter send a FORM FEED character (or whatever is necessary) to the printer. This is usually sufficient to have the printer immediately print any text remaining in its internal buffer. It is also useful to make sure each print job ends on a full sheet, so the next job does not start somewhere on the middle of the last page of the previous job.

The following replacement for the shell script `/usr/local/libexec/if-simple` prints a form feed after it sends the job to the printer:

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout. Ignores all filter arguments.
# Writes a form feed character (\f) after printing job.

/bin/cat && printf "\f" && exit 0
exit 2
```

It produced the „staircase effect.”

You got the following on paper:

```
! "#$%&'()*+,-./01234
    "#$%&'()*+,-./012345
        "#$%&'()*+,-./0123456
```

You have become another victim of the *staircase effect*, caused by conflicting interpretations of what characters should indicate a new line. UNIX style operating systems use a single character: ASCII code 10, the line feed (LF). MS-DOS, OS/2®, and others uses a pair of characters, ASCII code 10 *and* ASCII code 13 (the carriage return or CR). Many printers use the MS-DOS convention for representing new-lines.

When you print with FreeBSD, your text used just the line feed character. The printer, upon seeing a line feed character, advanced the paper one line, but maintained the same horizontal position on the page for the next character to print. That is what the carriage return is for: to move the location of the next character to print to the left edge of the paper.

Here is what FreeBSD wants your printer to do:

Printer received CR	Printer prints CR
Printer received LF	Printer prints CR + LF

Here are some ways to achieve this:

- Use the printer's configuration switches or control panel to alter its interpretation of these characters. Check your printer's manual to find out how to do this.

Notatka: If you boot your system into other operating systems besides FreeBSD, you may have to *reconfigure* the printer to use a an interpretation for CR and LF characters that those other operating systems use. You might prefer one of the other solutions, below.

- Have FreeBSD's serial line driver automatically convert LF to CR+LF. Of course, this works with printers on serial ports *only*. To enable this feature, use the `ms#` capability and set the `onlcr` mode in the `/etc/printcap` file for the printer.
- Send an *escape code* to the printer to have it temporarily treat LF characters differently. Consult your printer's manual for escape codes that your printer might support. When you find the proper escape code, modify the text filter to send the code first, then send the print job.

Here is an example text filter for printers that understand the Hewlett-Packard PCL escape codes. This filter makes the printer treat LF characters as a LF and CR; then it sends the job; then it sends a form feed to eject the last page of the job. It should work with nearly all Hewlett Packard printers.

```
#!/bin/sh
#
# hpif - Simple text input filter for lpd for HP-PCL based printers
# Installed in /usr/local/libexec/hpif
#
# Simply copies stdin to stdout. Ignores all filter arguments.
# Tells printer to treat LF as CR+LF. Ejects the page when done.

printf "\033&k2G" && cat && printf "\033&l0H" && exit 0
exit 2
```

Here is an example `/etc/printcap` from a host called `orchid`. It has a single printer attached to its first parallel port, a Hewlett Packard LaserJet 3Si named `teak`. It is using the above script as its text filter:

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:
```

It overprinted each line.

The printer never advanced a line. All of the lines of text were printed on top of each other on one line.

This problem is the „opposite” of the staircase effect, described above, and is much rarer. Somewhere, the LF characters that FreeBSD uses to end a line are being treated as CR characters to return the print location to the left edge of the paper, but not also down a line.

Use the printer's configuration switches or control panel to enforce the following interpretation of LF and CR characters:

Printer receives	Printer prints
CR	CR
LF	CR + LF

The printer lost characters.

While printing, the printer did not print a few characters in each line. The problem might have gotten worse as the printer ran, losing more and more characters.

The problem is that the printer cannot keep up with the speed at which the computer sends data over a serial line (this problem should not occur with printers on parallel ports). There are two ways to overcome the problem:

- If the printer supports XON/XOFF flow control, have FreeBSD use it by specifying the `ixon` mode in the `ms#` capability.
- If the printer supports carrier flow control, specify the `crtsets` mode in the `ms#` capability. Make sure the cable connecting the printer to the computer is correctly wired for carrier flow control.

It printed garbage.

The printer printed what appeared to be random garbage, but not the desired text.

This is usually another symptom of incorrect communications parameters with a serial printer. Double-check the bps rate in the `br` capability, and the parity setting in the `ms#` capability; make sure the printer is using the same settings as specified in the `/etc/printcap` file.

Nothing happened.

If nothing happened, the problem is probably within FreeBSD and not the hardware. Add the log file (`lf`) capability to the entry for the printer you are debugging in the `/etc/printcap` file. For example, here is the entry for `rattan`, with the `lf` capability:

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:\
    :lf=/var/log/rattan.log
```

Then, try printing again. Check the log file (in our example, `/var/log/rattan.log`) to see any error messages that might appear. Based on the messages you see, try to correct the problem.

If you do not specify a `lf` capability, **LPD** uses `/dev/console` as a default.

Rozdział 10. Linux Binary Compatibility

Restructured and parts updated by Jim Mock. Originally contributed by Brian N. Handy i Rich Murphey.

10.1. Synopsis

FreeBSD provides binary compatibility with several other UNIX like operating systems, including Linux. At this point, you may be asking yourself why exactly, does FreeBSD need to be able to run Linux binaries? The answer to that question is quite simple. Many companies and developers develop only for Linux, since it is the latest „hot thing” in the computing world. That leaves the rest of us FreeBSD users bugging these same companies and developers to put out native FreeBSD versions of their applications. The problem is, that most of these companies do not really realize how many people would use their product if there were FreeBSD versions too, and most continue to only develop for Linux. So what is a FreeBSD user to do? This is where the Linux binary compatibility of FreeBSD comes into play.

In a nutshell, the compatibility allows FreeBSD users to run about 90% of all Linux applications without modification. This includes applications such as **StarOffice**, the Linux version of **Netscape**, **Adobe Acrobat**, **RealPlayer®**, **VMware™**, **Oracle**, **WordPerfect®**, **Doom**, **Quake**, and more. It is also reported that in some situations, Linux binaries perform better on FreeBSD than they do under Linux.

There are, however, some Linux-specific operating system features that are not supported under FreeBSD. Linux binaries will not work on FreeBSD if they overly use i386 specific calls, such as enabling virtual 8086 mode.

After reading this chapter, you will know:

- How to enable Linux binary compatibility on your system.
- How to install additional Linux shared libraries.
- How to install Linux applications on your FreeBSD system.
- The implementation details of Linux compatibility in FreeBSD.

Before reading this chapter, you should:

- Know how to install additional third-party software (Rozdział 4).

10.2. Installation

Linux binary compatibility is not turned on by default. The easiest way to enable this functionality is to load the `linux` KLD object („Kernel Loadable object”). You can load this module by typing the following as `root`:

```
# kldload linux
```

If you would like Linux compatibility to always be enabled, then you should add the following line to `/etc/rc.conf`:

```
linux_enable="YES"
```

The `kldstat(8)` command can be used to verify that the KLD is loaded:

```
% kldstat
Id Refs Address      Size      Name
  1     2 0xc0100000 16bdb8    kernel
  7     1 0xc24db000 d000      linux.ko
```

If for some reason you do not want to or cannot load the KLD, then you may statically link Linux binary compatibility into the kernel by adding options `COMPAT_LINUX` to your kernel configuration file. Then install your new kernel as described in Rozdział 8.

10.2.1. Installing Linux Runtime Libraries

This can be done one of two ways, either by using the `linux_base` port, or by installing them manually.

10.2.1.1. Installing Using the `linux_base` Port

This is by far the easiest method to use when installing the runtime libraries. It is just like installing any other port from the Ports Collection (`/usr/ports/`). Simply do the following:

```
# cd /usr/ports/emulators/linux_base-fc4
# make install distclean
```

You should now have working Linux binary compatibility. Some programs may complain about incorrect minor versions of the system libraries. In general, however, this does not seem to be a problem.

Notatka: There may be multiple versions of the `emulators/linux_base` port available, corresponding to different versions of various Linux distributions. You should install the port most closely resembling the requirements of the Linux applications you would like to install.

10.2.1.2. Installing Libraries Manually

If you do not have the „ports” collection installed, you can install the libraries by hand instead. You will need the Linux shared libraries that the program depends on and the runtime linker. Also, you will need to create a „shadow root” directory, `/compat/linux`, for Linux libraries on your FreeBSD system. Any shared libraries opened by Linux programs run under FreeBSD will look in this tree first. So, if a Linux program loads, for example, `/lib/libc.so`, FreeBSD will first try to open `/compat/linux/lib/libc.so`, and if that does not exist, it will then try `/lib/libc.so`. Shared libraries should be installed in the shadow tree `/compat/linux/lib` rather than the paths that the Linux `ld.so` reports.

Generally, you will need to look for the shared libraries that Linux binaries depend on only the first few times that you install a Linux program on your FreeBSD system. After a while, you will have a sufficient set of Linux shared libraries on your system to be able to run newly imported Linux binaries without any extra work.

10.2.1.3. How to Install Additional Shared Libraries

What if you install the `linux_base` port and your application still complains about missing shared libraries? How do you know which shared libraries Linux binaries need, and where to get them? Basically, there are 2 possibilities (when following these instructions you will need to be `root` on your FreeBSD system).

If you have access to a Linux system, see what shared libraries the application needs, and copy them to your FreeBSD system. Look at the following example:

Let us assume you used FTP to get the Linux binary of **Doom**, and put it on a Linux system you have access to. You then can check which shared libraries it needs by running `ldd linuxdoom`, like so:

```
% ldd linuxdoom
libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.29
```

You would need to get all the files from the last column, and put them under `/compat/linux`, with the names in the first column as symbolic links pointing to them. This means you eventually have these files on your FreeBSD system:

```
/compat/linux/usr/X11/lib/libXt.so.3.1.0
/compat/linux/usr/X11/lib/libXt.so.3 -> libXt.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3 -> libX11.so.3.1.0
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

Notatka: Note that if you already have a Linux shared library with a matching major revision number to the first column of the `ldd` output, you will not need to copy the file named in the last column to your system, the one you already have should work. It is advisable to copy the shared library anyway if it is a newer version, though. You can remove the old one, as long as you make the symbolic link point to the new one. So, if you have these libraries on your system:

```
/compat/linux/lib/libc.so.4.6.27
/compat/linux/lib/libc.so.4 -> libc.so.4.6.27
```

and you find a new binary that claims to require a later version according to the output of `ldd`:

```
libc.so.4 (DLL Jump 4.5pl26) -> libc.so.4.6.29
```

If it is only one or two versions out of date in the trailing digit then do not worry about copying `/lib/libc.so.4.6.29` too, because the program should work fine with the slightly older version. However, if you like, you can decide to replace the `libc.so` anyway, and that should leave you with:

```
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

Notatka: The symbolic link mechanism is *only* needed for Linux binaries. The FreeBSD runtime linker takes care of looking for matching major revision numbers itself and you do not need to worry about it.

10.2.2. Installing Linux ELF Binaries

ELF binaries sometimes require an extra step of „branding”. If you attempt to run an unbranded ELF binary, you will get an error message like the following:

```
% ./my-linux-elf-binary
ELF binary type not known
Abort
```

To help the FreeBSD kernel distinguish between a FreeBSD ELF binary from a Linux binary, use the `brandelf(1)` utility.

```
% brandelf -t Linux my-linux-elf-binary
```

The GNU toolchain now places the appropriate branding information into ELF binaries automatically, so this step should become increasingly unnecessary in the future.

10.2.3. Configuring the Hostname Resolver

If DNS does not work or you get this message:

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

You will need to configure a `/compat/linux/etc/host.conf` file containing:

```
order hosts, bind
multi on
```

The order here specifies that `/etc/hosts` is searched first and DNS is searched second. When `/compat/linux/etc/host.conf` is not installed, Linux applications find FreeBSD's `/etc/host.conf` and complain about the incompatible FreeBSD syntax. You should remove `bind` if you have not configured a name server using the `/etc/resolv.conf` file.

10.3. Installing Mathematica®

Updated for Mathematica 5.X by Boris Hollas.

This document describes the process of installing the Linux version of **Mathematica 5.X** onto a FreeBSD system.

The Linux version of **Mathematica** or **Mathematica for Students** can be ordered directly from Wolfram at <http://www.wolfram.com/>.

10.3.1. Running the Mathematica Installer

First, you have to tell FreeBSD that **Mathematica**'s Linux binaries use the Linux ABI. The easiest way to do so is to set the default ELF brand to Linux for all unbranded binaries with the command:

```
# sysctl kern.fallback_elf_brand=3
```

This will make FreeBSD assume that unbranded ELF binaries use the Linux ABI and so you should be able to run the installer straight from the CDROM.

Now, copy the file `MathInstaller` to your hard drive:

```
# mount /cdrom
# cp /cdrom/Unix/Installers/Linux/MathInstaller /localdir/
```

and in this file, replace `/bin/sh` in the first line by `/compat/linux/bin/sh`. This makes sure that the installer is executed by the Linux version of `sh(1)`. Next, replace all occurrences of `Linux)` by `FreeBSD)` with a text editor or the script below in the next section. This tells the **Mathematica** installer, who calls `uname -s` to determine the operating system, to treat FreeBSD as a Linux-like operating system. Invoking `MathInstaller` will now install **Mathematica**.

10.3.2. Modifying the Mathematica Executables

The shell scripts that **Mathematica** created during installation have to be modified before you can use them. If you chose `/usr/local/bin` as the directory to place the **Mathematica** executables in, you will find symlinks in this directory to files called `math`, `mathematica`, `Mathematica`, and `MathKernel`. In each of these, replace `Linux)` by `FreeBSD)` with a text editor or the following shell script:

```
#!/bin/sh
cd /usr/local/bin
for i in math mathematica Mathematica MathKernel
do sed 's/Linux)/FreeBSD)/g' $i > $i.tmp
sed 's/\/bin\/sh/\/compat\/linux\/bin\/sh/g' $i.tmp > $i
rm $i.tmp
chmod a+x $i
done
```

10.3.3. Obtaining Your Mathematica Password

When you start **Mathematica** for the first time, you will be asked for a password. If you have not yet obtained a password from Wolfram, run the program `mathinfo` in the installation directory to obtain your „machine ID”. This machine ID is based solely on the MAC address of your first Ethernet card, so you cannot run your copy of **Mathematica** on different machines.

When you register with Wolfram, either by email, phone or fax, you will give them the „machine ID” and they will respond with a corresponding password consisting of groups of numbers.

10.3.4. Running the Mathematica Frontend over a Network

Mathematica uses some special fonts to display characters not present in any of the standard font sets (integrals, sums, Greek letters, etc.). The X protocol requires these fonts to be installed *locally*. This means you will have to copy these fonts from the CDROM or from a host with **Mathematica** installed to your local machine. These fonts are normally stored in `/cdrom/Unix/Files/SystemFiles/Fonts` on the CDROM, or `/usr/local/mathematica/SystemFiles/Fonts` on your hard drive. The actual fonts are in the subdirectories `Type1` and `X`. There are several ways to use them, as described below.

The first way is to copy them into one of the existing font directories in `/usr/X11R6/lib/X11/fonts`. This will require editing the `fonts.dir` file, adding the font names to it, and changing the number of fonts on the first line. Alternatively, you should also just be able to run `mkfontdir(1)` in the directory you have copied them to.

The second way to do this is to copy the directories to `/usr/X11R6/lib/X11/fonts`:

```
# cd /usr/X11R6/lib/X11/fonts
# mkdir X
# mkdir MathType1
# cd /cdrom/Unix/Files/SystemFiles/Fonts
# cp X/* /usr/X11R6/lib/X11/fonts/X
# cp Type1/* /usr/X11R6/lib/X11/fonts/MathType1
# cd /usr/X11R6/lib/X11/fonts/X
# mkfontdir
# cd ../MathType1
# mkfontdir
```

Now add the new font directories to your font path:

```
# xset fp+ /usr/X11R6/lib/X11/fonts/X
# xset fp+ /usr/X11R6/lib/X11/fonts/MathType1
# xset fp rehash
```

If you are using the **Xorg** server, you can have these font directories loaded automatically by adding them to your `xorg.conf` file.

Notatka: For **XFree86** servers, the configuration file is `XF86Config`.

If you *do not* already have a directory called `/usr/X11R6/lib/X11/fonts/Type1`, you can change the name of the `MathType1` directory in the example above to `Type1`.

10.4. Installing Maple™

Contributed by Aaron Kaplan. Thanks to Robert Getschmann.

Maple™ is a commercial mathematics program similar to **Mathematica**. You must purchase this software from <http://www.maplesoft.com/> and then register there for a license file. To install this software on FreeBSD, please follow these simple steps.

1. Execute the `INSTALL` shell script from the product distribution. Choose the „RedHat” option when prompted by the installation program. A typical installation directory might be `/usr/local/maple`.
2. If you have not done so, order a license for **Maple** from Maple Waterloo Software (<http://register.maplesoft.com/>) and copy it to `/usr/local/maple/license/license.dat`.
3. Install the **FLEXlm** license manager by running the `INSTALL_LIC` install shell script that comes with **Maple**. Specify the primary hostname for your machine for the license server.
4. Patch the `/usr/local/maple/bin/maple.system.type` file with the following:

```
----- snip -----
*** maple.system.type.orig      Sun Jul  8 16:35:33 2001
--- maple.system.type      Sun Jul  8 16:35:51 2001
*****
*** 72,77 ***
--- 72,78 ----
```

```

        # the IBM RS/6000 AIX case
        MAPLE_BIN="bin.IBM_RISC_UNIX"
        ;;
+   "FreeBSD" |\
    "Linux")
        # the Linux/x86 case
        # We have two Linux implementations, one for Red Hat and
        ----- snip end of patch -----

```

Please note that after the "FreeBSD" | \ no other whitespace should be present.

This patch instructs **Maple** to recognize „FreeBSD” as a type of Linux system. The bin/maple shell script calls the bin/maple.system.type shell script which in turn calls `uname -a` to find out the operating system name. Depending on the OS name it will find out which binaries to use.

5. Start the license server.

The following script, installed as `/usr/local/etc/rc.d/lmgrd.sh` is a convenient way to start up `lmgrd`:

```

----- snip -----

#!/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin
PATH=${PATH}:/usr/local/maple/bin:/usr/local/maple/FLEXlm/UNIX/LINUX
export PATH

LICENSE_FILE=/usr/local/maple/license/license.dat
LOG=/var/log/lmgrd.log

case "$1" in
start)
    lmgrd -c ${LICENSE_FILE} 2>> ${LOG} 1>&2
    echo -n " lmgrd"
    ;;
stop)
    lmgrd -c ${LICENSE_FILE} -x lmdown 2>> ${LOG} 1>&2
    ;;
*)
    echo "Usage: `basename $0` {start|stop}" 1>&2
    exit 64
    ;;
esac

exit 0
----- snip -----

```

6. Test-start **Maple**:

```

% cd /usr/local/maple/bin
% ./xmaple

```

You should be up and running. Make sure to write Maplesoft to let them know you would like a native FreeBSD version!

10.4.1. Common Pitfalls

- The **FLEXlm** license manager can be a difficult tool to work with. Additional documentation on the subject can be found at <http://www.globetrotter.com/>.
- `lmgrd` is known to be very picky about the license file and to core dump if there are any problems. A correct license file should look like this:

```
# =====
# License File for UNIX Installations ("Pointer File")
# =====
SERVER chillig ANY
#USE_SERVER
VENDOR maplelmg

FEATURE Maple maplelmg 2000.0831 permanent 1 XXXXXXXXXXXX \
    PLATFORMS=i86_r ISSUER="Waterloo Maple Inc." \
    ISSUED=11-may-2000 NOTICE=" Technische Universitat Wien" \
    SN=XXXXXXXXXX
```

Notatka: Serial number and key 'X'ed out. `chillig` is a hostname.

Editing the license file works as long as you do not touch the „FEATURE” line (which is protected by the license key).

10.5. Installing MATLAB®

Contributed by Dan Pelleg.

This document describes the process of installing the Linux version of **MATLAB® version 6.5** onto a FreeBSD system. It works quite well, with the exception of the **Java Virtual Machine™** (see Sekcja 10.5.3).

The Linux version of **MATLAB** can be ordered directly from The MathWorks at <http://www.mathworks.com>. Make sure you also get the license file or instructions how to create it. While you are there, let them know you would like a native FreeBSD version of their software.

10.5.1. Installing MATLAB

To install **MATLAB**, do the following:

1. Insert the installation CD and mount it. Become `root`, as recommended by the installation script. To start the installation script type:

```
# /compat/linux/bin/sh /cdrom/install
```

Podpowiedź: The installer is graphical. If you get errors about not being able to open a display, type `setenv HOME ~USER`, where `USER` is the user you did a `su(1)` as.

2. When asked for the **MATLAB** root directory, type: `/compat/linux/usr/local/matlab`.

Podpowiedź: For easier typing on the rest of the installation process, type this at your shell prompt: `set MATLAB=/compat/linux/usr/local/matlab`

3. Edit the license file as instructed when obtaining the **MATLAB** license.

Podpowiedź: You can prepare this file in advance using your favorite editor, and copy it to `$MATLAB/license.dat` before the installer asks you to edit it.

4. Complete the installation process.

At this point your **MATLAB** installation is complete. The following steps apply „glue” to connect it to your FreeBSD system.

10.5.2. License Manager Startup

1. Create symlinks for the license manager scripts:

```
# ln -s $MATLAB/etc/lmboot /usr/local/etc/lmboot_TMW
# ln -s $MATLAB/etc/lmdown /usr/local/etc/lmdown_TMW
```

2. Create a startup file at `/usr/local/etc/rc.d/flexlm.sh`. The example below is a modified version of the distributed `$MATLAB/etc/rc.lm.glnx86`. The changes are file locations, and startup of the license manager under Linux emulation.

```
#!/bin/sh
case "$1" in
  start)
    if [ -f /usr/local/etc/lmboot_TMW ]; then
      /compat/linux/bin/sh /usr/local/etc/lmboot_TMW -u username && echo 'MATLAB_lmgrd'
    fi
    ;;
  stop)
    if [ -f /usr/local/etc/lmdown_TMW ]; then
      /compat/linux/bin/sh /usr/local/etc/lmdown_TMW > /dev/null 2>&1
    fi
    ;;
  *)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac

exit 0
```

WAŻNE: The file must be made executable:

```
# chmod +x /usr/local/etc/rc.d/flexlm.sh
```

You must also replace `username` above with the name of a valid user on your system (and not `root`).

3. Start the license manager with the command:

```
# /usr/local/etc/rc.d/flexlm.sh start
```

10.5.3. Linking the Java Runtime Environment

Change the **Java** Runtime Environment (JRE) link to one working under FreeBSD:

```
# cd $MATLAB/sys/java/jre/glnx86/
# unlink jre; ln -s ./jre1.1.8 ./jre
```

10.5.4. Creating a MATLAB Startup Script

1. Place the following startup script in `/usr/local/bin/matlab`:

```
#!/bin/sh
/compat/linux/bin/sh /compat/linux/usr/local/matlab/bin/matlab "$@"
```

2. Then type the command `chmod +x /usr/local/bin/matlab`.

Podpowiedź: Depending on your version of `emulators/linux_base`, you may run into errors when running this script. To avoid that, edit the file `/compat/linux/usr/local/matlab/bin/matlab`, and change the line that says:

```
if [ `expr "$lsCmd" : '.*->.*'` -ne 0 ]; then
```

(in version 13.0.1 it is on line 410) to this line:

```
if test -L $newbase; then
```

10.5.5. Creating a MATLAB Shutdown Script

The following is needed to solve a problem with MATLAB not exiting correctly.

1. Create a file `$MATLAB/toolbox/local/finish.m`, and in it put the single line:

```
! $MATLAB/bin/finish.sh
```

Notatka: The `$MATLAB` is literal.

Podpowiedź: In the same directory, you will find the files `finishsav.m` and `finishdlg.m`, which let you save your workspace before quitting. If you use either of them, insert the line above immediately after the `save` command.

2. Create a file `$MATLAB/bin/finish.sh`, which will contain the following:

```
#!/usr/compat/linux/bin/sh
(sleep 5; killall -1 matlab_helper) &
exit 0
```

3. Make the file executable:

```
# chmod +x $MATLAB/bin/finish.sh
```

10.5.6. Using MATLAB

At this point you are ready to type `matlab` and start using it.

10.6. Installing Oracle®

Contributed by Marcel Moolenaar.

10.6.1. Preface

This document describes the process of installing **Oracle 8.0.5** and **Oracle 8.0.5.1 Enterprise Edition** for Linux onto a FreeBSD machine.

10.6.2. Installing the Linux Environment

Make sure you have both `emulators/linux_base` and `devel/linux_devtools` from the Ports Collection installed. If you run into difficulties with these ports, you may have to use the packages or older versions available in the Ports Collection.

If you want to run the intelligent agent, you will also need to install the Red Hat Tcl package:

`tcl-8.0.3-20.i386.rpm`. The general command for installing packages with the official **RPM** port (`archivers/rpm`) is:

```
# rpm -i --ignoreos --root /compat/linux --dbpath /var/lib/rpm package
```

Installation of the *package* should not generate any errors.

10.6.3. Creating the Oracle Environment

Before you can install **Oracle**, you need to set up a proper environment. This document only describes what to do *specifically* to run **Oracle** for Linux on FreeBSD, not what has been described in the **Oracle** installation guide.

10.6.3.1. Kernel Tuning

As described in the **Oracle** installation guide, you need to set the maximum size of shared memory. Do not use SHMMAX under FreeBSD. SHMMAX is merely calculated out of SHMAXPGS and PGSIZE. Therefore define SHMAXPGS. All other options can be used as described in the guide. For example:

```
options SHMAXPGS=10000
options SHMMNI=100
options SHMSEG=10
options SEMMNS=200
options SEMMNI=70
options SEMMSL=61
```

Set these options to suit your intended use of **Oracle**.

Also, make sure you have the following options in your kernel configuration file:

```
options SYSVSHM #SysV shared memory
options SYSVSEM #SysV semaphores
options SYSVMSG #SysV interprocess communication
```

10.6.3.2. Oracle Account

Create an `oracle` account just as you would create any other account. The `oracle` account is special only that you need to give it a Linux shell. Add `/compat/linux/bin/bash` to `/etc/shells` and set the shell for the `oracle` account to `/compat/linux/bin/bash`.

10.6.3.3. Environment

Besides the normal **Oracle** variables, such as `ORACLE_HOME` and `ORACLE_SID` you must set the following environment variables:

Variable	Value
<code>LD_LIBRARY_PATH</code>	<code>\$ORACLE_HOME/lib</code>
<code>CLASSPATH</code>	<code>\$ORACLE_HOME/jdbc/lib/classes111.zip</code>
<code>PATH</code>	<code>/compat/linux/bin /compat/linux/sbin</code> <code>/compat/linux/usr/bin /compat/linux/usr/sbin /bin /sbin</code> <code>/usr/bin /usr/sbin /usr/local/bin \$ORACLE_HOME/bin</code>

It is advised to set all the environment variables in `.profile`. A complete example is:

```
ORACLE_BASE=/oracle; export ORACLE_BASE
ORACLE_HOME=/oracle; export ORACLE_HOME
LD_LIBRARY_PATH=$ORACLE_HOME/lib
export LD_LIBRARY_PATH
ORACLE_SID=ORCL; export ORACLE_SID
ORACLE_TERM=386x; export ORACLE_TERM
CLASSPATH=$ORACLE_HOME/jdbc/lib/classes111.zip
export CLASSPATH
PATH=/compat/linux/bin:/compat/linux/sbin:/compat/linux/usr/bin
```

```
PATH=$PATH:/compat/linux/usr/sbin:/bin:/sbin:/usr/bin:/usr/sbin
PATH=$PATH:/usr/local/bin:$ORACLE_HOME/bin
export PATH
```

10.6.4. Installing Oracle

Due to a slight inconsistency in the Linux emulator, you need to create a directory named `.oracle` in `/var/tmp` before you start the installer. Let it be owned by the `oracle` user. You should be able to install **Oracle** without any problems. If you have problems, check your **Oracle** distribution and/or configuration first! After you have installed **Oracle**, apply the patches described in the next two subsections.

A frequent problem is that the TCP protocol adapter is not installed right. As a consequence, you cannot start any TCP listeners. The following actions help solve this problem:

```
# cd $ORACLE_HOME/network/lib
# make -f ins_network.mk ntcontab.o
# cd $ORACLE_HOME/lib
# ar r libnetwork.a ntcontab.o
# cd $ORACLE_HOME/network/lib
# make -f ins_network.mk install
```

Do not forget to run `root.sh` again!

10.6.4.1. Patching root.sh

When installing **Oracle**, some actions, which need to be performed as `root`, are recorded in a shell script called `root.sh`. This script is written in the `orainst` directory. Apply the following patch to `root.sh`, to have it use to proper location of `chown` or alternatively run the script under a Linux native shell.

```
*** orainst/root.sh.orig Tue Oct 6 21:57:33 1998
--- orainst/root.sh Mon Dec 28 15:58:53 1998
*****
*** 31,37 ****
# This is the default value for CHOWN
# It will redefined later in this script for those ports
# which have it conditionally defined in ss_install.h
! CHOWN=/bin/chown
#
# Define variables to be used in this script
--- 31,37 ----
# This is the default value for CHOWN
# It will redefined later in this script for those ports
# which have it conditionally defined in ss_install.h
! CHOWN=/usr/sbin/chown
#
# Define variables to be used in this script
```

When you do not install **Oracle** from CD, you can patch the source for `root.sh`. It is called `rthd.sh` and is located in the `orainst` directory in the source tree.

10.6.4.2. Patching gencintsh

The script `gencintsh` is used to create a single shared client library. It is used when building the demos. Apply the following patch to comment out the definition of `PATH`:

```
*** bin/gencintsh.orig Wed Sep 30 07:37:19 1998
--- bin/gencintsh Tue Dec 22 15:36:49 1998
*****
*** 32,38 ****
#
# Explicit path to ensure that we're using the correct commands
#PATH=/usr/bin:/usr/ccs/bin export PATH
! PATH=/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin export PATH
#
# each product MUST provide a $PRODUCT/admin/shrept.lst
--- 32,38 ----
#
# Explicit path to ensure that we're using the correct commands
#PATH=/usr/bin:/usr/ccs/bin export PATH
! #PATH=/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin export PATH
#
# each product MUST provide a $PRODUCT/admin/shrept.lst
```

10.6.5. Running Oracle

When you have followed the instructions, you should be able to run **Oracle** as if it was run on Linux itself.

10.7. Installing SAP® R/3®

Contributed by Holger Kipp. Original version converted to SGML by Valentino Vaschetto.

Installations of **SAP Systems** using **FreeBSD** will not be supported by the **SAP support team** — they only offer support for certified platforms.

10.7.1. Preface

This document describes a possible way of installing a **SAP R/3 System** with **Oracle Database** for Linux onto a **FreeBSD** machine, including the installation of **FreeBSD** and **Oracle**. Two different configurations will be described:

- **SAP R/3 4.6B (IDES)** with **Oracle 8.0.5** on **FreeBSD 4.3-STABLE**
- **SAP R/3 4.6C** with **Oracle 8.1.7** on **FreeBSD 4.5-STABLE**

Even though this document tries to describe all important steps in a greater detail, it is not intended as a replacement for the **Oracle** and **SAP R/3** installation guides.

Please see the documentation that comes with the **SAP R/3 Linux** edition for **SAP** and **Oracle** specific questions, as well as resources from **Oracle** and **SAP OSS**.

10.7.2. Software

The following CD-ROMs have been used for **SAP** installations:

10.7.2.1. SAP R/3 4.6B, Oracle 8.0.5

Name	Number	Description
KERNEL	51009113	SAP Kernel Oracle / Installation / AIX, Linux, Solaris
RDBMS	51007558	Oracle / RDBMS 8.0.5.X / Linux
EXPORT1	51010208	IDES / DB-Export / Disc 1 of 6
EXPORT2	51010209	IDES / DB-Export / Disc 2 of 6
EXPORT3	51010210	IDES / DB-Export / Disc 3 of 6
EXPORT4	51010211	IDES / DB-Export / Disc 4 of 6
EXPORT5	51010212	IDES / DB-Export / Disc 5 of 6
EXPORT6	51010213	IDES / DB-Export / Disc 6 of 6

Additionally, we used the **Oracle 8 Server** (Pre-production version 8.0.5 for Linux, Kernel Version 2.0.33) CD which is not really necessary, and FreeBSD 4.3-STABLE (it was only a few days past 4.3 RELEASE).

10.7.2.2. SAP R/3 4.6C SR2, Oracle 8.1.7

Name	Number	Description
KERNEL	51014004	SAP Kernel Oracle / SAP Kernel Version 4.6D / DEC, Linux
RDBMS	51012930	Oracle 8.1.7/ RDBMS / Linux
EXPORT1	51013953	Release 4.6C SR2 / Export / Disc 1 of 4
EXPORT1	51013953	Release 4.6C SR2 / Export / Disc 2 of 4
EXPORT1	51013953	Release 4.6C SR2 / Export / Disc 3 of 4
EXPORT1	51013953	Release 4.6C SR2 / Export / Disc 4 of 4
LANG1	51013954	Release 4.6C SR2 / Language / DE, EN, FR / Disc 1 of 3

Depending on the languages you would like to install, additional language CDs might be necessary. Here we are just using DE and EN, so the first language CD is the only one needed. As a little note, the numbers for all four EXPORT CDs are identical. All three language CDs also have the same number (this is different from the 4.6B IDES release CD numbering). At the time of writing this installation is running on FreeBSD 4.5-STABLE (20.03.2002).

10.7.3. SAP Notes

The following notes should be read before installing **SAP R/3** and proved to be useful during installation:

10.7.3.1. SAP R/3 4.6B, Oracle 8.0.5

Number	Title
0171356	SAP Software on Linux: Essential Comments
0201147	INST: 4.6C R/3 Inst. on UNIX - Oracle
0373203	Update / Migration Oracle 8.0.5 --> 8.0.6/8.1.6 LINUX
0072984	Release of Digital UNIX 4.0B for Oracle
0130581	R3SETUP step DIPGNTAB terminates
0144978	Your system has not been installed correctly
0162266	Questions and tips for R3SETUP on Windows NT / W2K

10.7.3.2. SAP R/3 4.6C, Oracle 8.1.7

Number	Title
0015023	Initializing table TCPDB (RSXP0004) (EBCDIC)
0045619	R/3 with several languages or typefaces
0171356	SAP Software on Linux: Essential Comments
0195603	RedHat 6.1 Enterprise version: Known problems
0212876	The new archiving tool SAPCAR
0300900	Linux: Released DELL Hardware
0377187	RedHat 6.2: important remarks
0387074	INST: R/3 4.6C SR2 Installation on UNIX
0387077	INST: R/3 4.6C SR2 Inst. on UNIX - Oracle
0387078	SAP Software on UNIX: OS Dependencies 4.6C SR2

10.7.4. Hardware Requirements

The following equipment is sufficient for the installation of a **SAP R/3 System**. For production use, a more exact sizing is of course needed:

Component	4.6B	4.6C
Processor	2 x 800MHz Pentium III	2 x 800MHz Pentium III
Memory	1GB ECC	2GB ECC
Hard Disk Space	50-60GB (IDES)	50-60GB (IDES)

For use in production, Xeon Processors with large cache, high-speed disk access (SCSI, RAID hardware controller), USV and ECC-RAM is recommended. The large amount of hard disk space is due to the preconfigured IDEs

System, which creates 27 GB of database files during installation. This space is also sufficient for initial production systems and application data.

10.7.4.1. SAP R/3 4.6B, Oracle 8.0.5

The following off-the-shelf hardware was used: a dual processor board with 2 800 MHz Pentium III processors, Adaptec® 29160 Ultra160 SCSI adapter (for accessing a 40/80 GB DLT tape drive and CDROM), Mylex® AcceleRAID™ (2 channels, firmware 6.00-1-00 with 32 MB RAM). To the Mylex RAID controller are attached two 17 GB hard disks (mirrored) and four 36 GB hard disks (RAID level 5).

10.7.4.2. SAP R/3 4.6C, Oracle 8.1.7

For this installation a Dell™ PowerEdge™ 2500 was used: a dual processor board with two 1000 MHz Pentium III processors (256 kB Cache), 2 GB PC133 ECC SDRAM, PERC/3 DC PCI RAID Controller with 128 MB, and an EIDE DVD-ROM drive. To the RAID controller are attached two 18 GB hard disks (mirrored) and four 36 GB hard disks (RAID level 5).

10.7.5. Installation of FreeBSD

First you have to install FreeBSD. There are several ways to do this, for more information read the Sekcja 2.13.

10.7.5.1. Disk Layout

To keep it simple, the same disk layout both for the **SAP R/3 46B** and **SAP R/3 46C SR2** installation was used. Only the device names changed, as the installations were on different hardware (`/dev/da` and `/dev/amr` respectively, so if using an AMI MegaRAID®, one will see `/dev/amr0s1a` instead of `/dev/da0s1a`):

File system	Size (1k-blocks)	Size (GB)	Mounted on
<code>/dev/da0s1a</code>	1.016.303	1	<code>/</code>
<code>/dev/da0s1b</code>		6	<code>swap</code>
<code>/dev/da0s1e</code>	2.032.623	2	<code>/var</code>
<code>/dev/da0s1f</code>	8.205.339	8	<code>/usr</code>
<code>/dev/da1s1e</code>	45.734.361	45	<code>/compat/linux/oracle</code>
<code>/dev/da1s1f</code>	2.032.623	2	<code>/compat/linux/sapmnt</code>
<code>/dev/da1s1g</code>	2.032.623	2	<code>/compat/linux/usr/sap</code>

Configure and initialize the two logical drives with the Mylex or PERC/3 RAID software beforehand. The software can be started during the BIOS boot phase.

Please note that this disk layout differs slightly from the SAP recommendations, as SAP suggests mounting the **Oracle** subdirectories (and some others) separately — we decided to just create them as real subdirectories for simplicity.

10.7.5.2. make world and a New Kernel

Download the latest -STABLE sources. Rebuild world and your custom kernel after configuring your kernel

configuration file. Here you should also include the kernel parameters which are required for both **SAP R/3** and **Oracle**.

10.7.6. Installing the Linux Environment

10.7.6.1. Installing the Linux Base System

First the `linux_base` port needs to be installed (as `root`):

```
# cd /usr/ports/emulators/linux_base
# make install distclean
```

10.7.6.2. Installing Linux Development Environment

The Linux development environment is needed, if you want to install **Oracle** on FreeBSD according to the Sekcja 10.6:

```
# cd /usr/ports/devel/linux_devtools
# make install distclean
```

The Linux development environment has only been installed for the **SAP R/3 46B IDES** installation. It is not needed, if the **Oracle DB** is not relinked on the FreeBSD system. This is the case if you are using the **Oracle** tarball from a Linux system.

10.7.6.3. Installing the Necessary RPMs

To start the `R3SETUP` program, PAM support is needed. During the first **SAP** Installation on FreeBSD 4.3-STABLE we tried to install PAM with all the required packages and finally forced the installation of the PAM package, which worked. For **SAP R/3 4.6C SR2** we directly forced the installation of the PAM RPM, which also works, so it seems the dependent packages are not needed:

```
# rpm -i --ignoreos --nodeps --root /compat/linux --dbpath /var/lib/rpm \
pam-0.68-7.i386.rpm
```

For **Oracle 8.0.5** to run the intelligent agent, we also had to install the RedHat Tcl package `tcl-8.0.5-30.i386.rpm` (otherwise the relinking during **Oracle** installation will not work). There are some other issues regarding relinking of **Oracle**, but that is a **Oracle** Linux issue, not FreeBSD specific.

10.7.6.4. Some Additional Hints

It might also be a good idea to add `linprocfs` to `/etc/fstab`, for more information, see the `linprocfs(5)` manual page. Another parameter to set is `kern.fallback_elf_brand=3` which is done in the file `/etc/sysctl.conf`.

10.7.7. Creating the SAP R/3 Environment

10.7.7.1. Creating the Necessary File Systems and Mountpoints

For a simple installation, it is sufficient to create the following file systems:

mount point	size in GB
/compat/linux/oracle	45 GB
/compat/linux/sapmnt	2 GB
/compat/linux/usr/sap	2 GB

It is also necessary to create some links. Otherwise the **SAP** Installer will complain, as it is checking the created links:

```
# ln -s /compat/linux/oracle /oracle
# ln -s /compat/linux/sapmnt /sapmnt
# ln -s /compat/linux/usr/sap /usr/sap
```

Possible error message during installation (here with System *PRD* and the **SAP R/3 4.6C SR2** installation):

```
INFO 2002-03-19 16:45:36 R3LINKS_IND_IND SyLinkCreate:200
    Checking existence of symbolic link /usr/sap/PRD/SYS/exe/dbg to
    /sapmnt/PRD/exe. Creating if it does not exist...

WARNING 2002-03-19 16:45:36 R3LINKS_IND_IND SyLinkCreate:400
    Link /usr/sap/PRD/SYS/exe/dbg exists but it points to file
    /compat/linux/sapmnt/PRD/exe instead of /sapmnt/PRD/exe. The
    program cannot go on as long as this link exists at this
    location. Move the link to another location.

ERROR 2002-03-19 16:45:36 R3LINKS_IND_IND Ins_SetupLinks:0
    can not setup link '/usr/sap/PRD/SYS/exe/dbg' with content
    '/sapmnt/PRD/exe'
```

10.7.7.2. Creating Users and Directories

SAP R/3 needs two users and three groups. The user names depend on the **SAP** system ID (SID) which consists of three letters. Some of these SIDs are reserved by **SAP** (for example *SAP* and *NIX*. For a complete list please see the **SAP** documentation). For the *IDES* installation we used *IDS*, for the *4.6C SR2* installation *PRD*, as that system is intended for production use. We have therefore the following groups (group IDs might differ, these are just the values we used with our installation):

group ID	group name	description
100	dba	Data Base Administrator
101	sapsys	SAP System
102	oper	Data Base Operator

For a default **Oracle** installation, only group *dba* is used. As *oper* group, one also uses group *dba* (see **Oracle** and **SAP** documentation for further information).

We also need the following users:

user ID	user name	generic name	group	additional groups	description
1000	idsadm/prdadm	<i>sidadm</i>	sapsys	oper	SAP Administrator
1002	oraid/oraprd	<i>orasid</i>	dba	oper	Oracle Administrator

Adding the users with `adduser(8)` requires the following (please note shell and home directory) entries for „SAP Administrator”:

```
Name: sidadm
Password: *****
Fullname: SAP Administrator SID
Uid: 1000
Gid: 101 (sapsys)
Class:
Groups: sapsys dba
HOME: /home/sidadm
Shell: bash (/compat/linux/bin/bash)
```

and for „Oracle Administrator”:

```
Name: orasid
Password: *****
Fullname: Oracle Administrator SID
Uid: 1002
Gid: 100 (dba)
Class:
Groups: dba
HOME: /oracle/sid
Shell: bash (/compat/linux/bin/bash)
```

This should also include group `oper` in case you are using both groups `dba` and `oper`.

10.7.7.3. Creating Directories

These directories are usually created as separate file systems. This depends entirely on your requirements. We choose to create them as simple directories, as they are all located on the same RAID 5 anyway:

First we will set owners and rights of some directories (as user `root`):

```
# chmod 775 /oracle
# chmod 777 /sapmnt
# chown root:dba /oracle
# chown sidadm:sapsys /compat/linux/usr/sap
# chmod 775 /compat/linux/usr/sap
```

Second we will create directories as user `orasid`. These will all be subdirectories of `/oracle/SID`:

```
# su - orasid
```

```
# cd /oracle/SID
# mkdir mirrlogA mirrlogB origlogA origlogB
# mkdir sapdata1 sapdata2 sapdata3 sapdata4 sapdata5 sapdata6
# mkdir saparch sapreorg
# exit
```

For the **Oracle 8.1.7** installation some additional directories are needed:

```
# su - orasid
# cd /oracle
# mkdir 805_32
# mkdir client stage
# mkdir client/80x_32
# mkdir stage/817_32
# cd /oracle/SID
# mkdir 817_32
```

Notatka: The directory `client/80x_32` is used with exactly this name. Do not replace the `x` with some number or anything.

In the third step we create directories as user `sidadm`:

```
# su - sidadm
# cd /usr/sap
# mkdir SID
# mkdir trans
# exit
```

10.7.7.4. Entries in `/etc/services`

SAP R/3 requires some entries in file `/etc/services`, which will not be set correctly during installation under FreeBSD. Please add the following entries (you need at least those entries corresponding to the instance number — in this case, 00. It will do no harm adding all entries from 00 to 99 for `dp`, `gw`, `sp` and `ms`). If you are going to use a **SAProuter** or need to access **SAP OSS**, you also need 99, as port 3299 is usually used for the **SAProuter** process on the target system:

```
sapdp00    3200/tcp # SAP Dispatcher.          3200 + Instance-Number
sapgw00    3300/tcp # SAP Gateway.              3300 + Instance-Number
sapsp00    3400/tcp #                          3400 + Instance-Number
sapms00    3500/tcp #                          3500 + Instance-Number
sapmsSID    3600/tcp # SAP Message Server.    3600 + Instance-Number
sapgw00s    4800/tcp # SAP Secure Gateway    4800 + Instance-Number
```

10.7.7.5. Necessary Locales

SAP requires at least two locales that are not part of the default RedHat installation. SAP offers the required RPMs as download from their FTP server (which is only accessible if you are a customer with OSS access). See note 0171356 for a list of RPMs you need.

It is also possible to just create appropriate links (for example from *de_DE* and *en_US*), but we would not recommend this for a production system (so far it worked with the IDES system without any problems, though). The following locales are needed:

```
de_DE.ISO-8859-1
en_US.ISO-8859-1
```

Create the links like this:

```
# cd /compat/linux/usr/share/locale
# ln -s de_DE de_DE.ISO-8859-1
# ln -s en_US en_US.ISO-8859-1
```

If they are not present, there will be some problems during the installation. If these are then subsequently ignored (by setting the STATUS of the offending steps to OK in file CENTRDB.R3S), it will be impossible to log onto the **SAP** system without some additional effort.

10.7.7.6. Kernel Tuning

SAP R/3 systems need a lot of resources. We therefore added the following parameters to the kernel configuration file:

```
# Set these for memory pigs (SAP and Oracle):
options MAXDSIZ="(1024*1024*1024)"
options DFLDSIZ="(1024*1024*1024)"
# System V options needed.
options SYSVSHM #SYSV-style shared memory
options SHMMAXPGS=262144 #max amount of shared mem. pages
#options SHMMAXPGS=393216 #use this for the 46C inst.parameters
options SHMMNI=256 #max number of shared memory ident if.
options SHMSEG=100 #max shared mem.segs per process
options SYSVMSG #SYSV-style message queues
options MSGSEG=32767 #max num. of mes.segments in system
options MSGSSZ=32 #size of msg-seg. MUST be power of 2
options MSGMNB=65535 #max char. per message queue
options MSGTQL=2046 #max amount of msgs in system
options SYSVSEM #SYSV-style semaphores
options SEMMNU=256 #number of semaphore UNDO structures
options SEMMNS=1024 #number of semaphores in system
options SEMMNI=520 #number of semaphore identifiers
options SEMUME=100          #number of UNDO keys
```

The minimum values are specified in the documentation that comes from SAP. As there is no description for Linux, see the HP-UX section (32-bit) for further information. As the system for the 4.6C SR2 installation has more main memory, the shared segments can be larger both for **SAP** and **Oracle**, therefore choose a larger number of shared memory pages.

Notatka: With the default installation of FreeBSD on i386, leave `MAXDSIZ` and `DFLDSIZ` at 1 GB maximum. Otherwise, strange errors like `ORA-27102: out of memory` and `Linux Error: 12: Cannot allocate memory` might happen.

10.7.8. Installing SAP R/3

10.7.8.1. Preparing SAP CDROMs

There are many CDROMs to mount and unmount during the installation. Assuming you have enough CDROM drives, you can just mount them all. We decided to copy the CDROMs contents to corresponding directories:

```
/oracle/SID/sapreorg/cd-name
```

where *cd-name* was one of KERNEL, RDBMS, EXPORT1, EXPORT2, EXPORT3, EXPORT4, EXPORT5 and EXPORT6 for the 4.6B/IDES installation, and KERNEL, RDBMS, DISK1, DISK2, DISK3, DISK4 and LANG for the 4.6C SR2 installation. All the filenames on the mounted CDs should be in capital letters, otherwise use the `-g` option for mounting. So use the following commands:

```
# mount_cd9660 -g /dev/cd0a /mnt
# cp -R /mnt/* /oracle/SID/sapreorg/cd-name
# umount /mnt
```

10.7.8.2. Running the Installation Script

First you have to prepare an `install` directory:

```
# cd /oracle/SID/sapreorg
# mkdir install
# cd install
```

Then the installation script is started, which will copy nearly all the relevant files into the `install` directory:

```
# /oracle/SID/sapreorg/KERNEL/UNIX/INSTTOOL.SH
```

The IDES installation (4.6B) comes with a fully customized SAP R/3 demonstration system, so there are six instead of just three EXPORT CDs. At this point the installation template `CENTRDB.R3S` is for installing a standard central instance (**R/3** and database), not the IDES central instance, so one needs to copy the corresponding `CENTRDB.R3S` from the `EXPORT1` directory, otherwise `R3SETUP` will only ask for three EXPORT CDs.

The newer **SAP 4.6C SR2** release comes with four EXPORT CDs. The parameter file that controls the installation steps is `CENTRAL.R3S`. Contrary to earlier releases there are no separate installation templates for a central instance with or without database. **SAP** is using a separate template for database installation. To restart the installation later it is however sufficient to restart with the original file.

During and after installation, **SAP** requires `hostname` to return the computer name only, not the fully qualified domain name. So either set the `hostname` accordingly, or set an alias with `alias hostname='hostname -s'` for both `orasid` and `sidadm` (and for `root` at least during installation steps performed as `root`). It is also possible to adjust the installed `.profile` and `.login` files of both users that are installed during **SAP** installation.

10.7.8.3. Start R3SETUP 4.6B

Make sure `LD_LIBRARY_PATH` is set correctly:

```
# export LD_LIBRARY_PATH=/oracle/IDS/lib:/sapmnt/IDS/exe:/oracle/805_32/lib
```

Start R3SETUP as root from installation directory:

```
# cd /oracle/IDS/sapreorg/install
# ./R3SETUP -f CENTRDB.R3S
```

The script then asks some questions (defaults in brackets, followed by actual input):

Question	Default	Input
Enter SAP System ID	[C11]	IDS Enter
Enter SAP Instance Number	[00]	Enter
Enter SAPMOUNT Directory	[/sapmnt]	Enter
Enter name of SAP central host	[troubadix.domain.de]	Enter
Enter name of SAP db host	[troubadix]	Enter
Select character set	[1] (WE8DEC)	Enter
Enter Oracle server version (1) Oracle 8.0.5, (2) Oracle 8.0.6, (3) Oracle 8.1.5, (4) Oracle 8.1.6		1 Enter
Extract Oracle Client archive	[1] (Yes, extract)	Enter
Enter path to KERNEL CD	[/sapcd]	/oracle/IDS/sapreorg/KERNEL
Enter path to RDBMS CD	[/sapcd]	/oracle/IDS/sapreorg/RDBMS
Enter path to EXPORT1 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT1
Directory to copy EXPORT1 CD	[/oracle/IDS/sapreorg/CD4_DIR]	Enter
Enter path to EXPORT2 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT2
Directory to copy EXPORT2 CD	[/oracle/IDS/sapreorg/CD5_DIR]	Enter
Enter path to EXPORT3 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT3
Directory to copy EXPORT3 CD	[/oracle/IDS/sapreorg/CD6_DIR]	Enter
Enter path to EXPORT4 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT4
Directory to copy EXPORT4 CD	[/oracle/IDS/sapreorg/CD7_DIR]	Enter
Enter path to EXPORT5 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT5
Directory to copy EXPORT5 CD	[/oracle/IDS/sapreorg/CD8_DIR]	Enter
Enter path to EXPORT6 CD	[/sapcd]	/oracle/IDS/sapreorg/EXPORT6
Directory to copy EXPORT6 CD	[/oracle/IDS/sapreorg/CD9_DIR]	Enter
Enter amount of RAM for SAP + DB		850 Enter (in Megabytes)
Service Entry Message Server	[3600]	Enter
Enter Group-ID of sapsys	[101]	Enter
Enter Group-ID of oper	[102]	Enter
Enter Group-ID of dba	[100]	Enter
Enter User-ID of <i>sidadm</i>	[1000]	Enter
Enter User-ID of <i>orasid</i>	[1002]	Enter
Number of parallel procs	[2]	Enter

If you had not copied the CDs to the different locations, then the **SAP** installer cannot find the CD needed (identified by the LABEL. ASC file on the CD) and would then ask you to insert and mount the CD and confirm or enter the mount path.

The `CENTRDB.R3S` might not be error free. In our case, it requested `EXPORT4 CD` again but indicated the correct key (`6_LOCATION`, then `7_LOCATION` etc.), so one can just continue with entering the correct values.

Apart from some problems mentioned below, everything should go straight through up to the point where the Oracle database software needs to be installed.

10.7.8.4. Start `R3SETUP 4.6C SR2`

Make sure `LD_LIBRARY_PATH` is set correctly. This is a different value from the 4.6B installation with **Oracle 8.0.5**:

```
# export LD_LIBRARY_PATH=/sapmnt/PRD/exe:/oracle/PRD/817_32/lib
```

Start `R3SETUP` as user `root` from installation directory:

```
# cd /oracle/PRD/sapreorg/install
# ./R3SETUP -f CENTRAL.R3S
```

The script then asks some questions (defaults in brackets, followed by actual input):

Question	Default	Input
Enter SAP System ID	[C11]	PRDEnter
Enter SAP Instance Number	[00]	Enter
Enter SAPMOUNT Directory	[/sapmnt]	Enter
Enter name of SAP central host	[majestix]	Enter
Enter Database System ID	[PRD]	PRDEnter
Enter name of SAP db host	[majestix]	Enter
Select character set	[1] (WE8DEC)	Enter
Enter Oracle server version (2) Oracle 8.1.7		2Enter
Extract Oracle Client archive	[1] (Yes, extract)	Enter
Enter path to KERNEL CD	[/sapcd]	/oracle/PRD/sapreorg/KERNEL
Enter amount of RAM for SAP + DB	2044	1800Enter (in Megabytes)
Service Entry Message Server	[3600]	Enter
Enter Group-ID of sapsys	[100]	Enter
Enter Group-ID of oper	[101]	Enter
Enter Group-ID of dba	[102]	Enter
Enter User-ID of oraprd	[1002]	Enter
Enter User-ID of prdadm	[1000]	Enter
LDAP support		3Enter (no support)
Installation step completed	[1] (continue)	Enter
Choose installation service	[1] (DB inst,file)	Enter

So far, creation of users gives an error during installation in phases `OSUSERDBSID_IND_ORA` (for creating user `orasid`) and `OSUSERSIDADM_IND_ORA` (creating user `sidadm`).

Apart from some problems mentioned below, everything should go straight through up to the point where the Oracle database software needs to be installed.

10.7.9. Installing Oracle 8.0.5

Please see the corresponding SAP Notes and Oracle Readmes regarding Linux and **Oracle DB** for possible problems. Most if not all problems stem from incompatible libraries.

For more information on installing **Oracle**, refer to the Installing Oracle chapter.

10.7.9.1. Installing the Oracle 8.0.5 with `orainst`

If **Oracle 8.0.5** is to be used, some additional libraries are needed for successfully relinking, as **Oracle 8.0.5** was linked with an old glibc (RedHat 6.0), but RedHat 6.1 already uses a new glibc. So you have to install the following additional packages to ensure that linking will work:

```
compat-libs-5.2-2.i386.rpm
compat-glibc-5.2-2.0.7.2.i386.rpm
compat-egcs-5.2-1.0.3a.1.i386.rpm
compat-egcs-c++-5.2-1.0.3a.1.i386.rpm
compat-binutils-5.2-2.9.1.0.23.1.i386.rpm
```

See the corresponding SAP Notes or Oracle Readmes for further information. If this is no option (at the time of installation we did not have enough time to check this), one could use the original binaries, or use the relinked binaries from an original RedHat system.

For compiling the intelligent agent, the RedHat Tcl package must be installed. If you cannot get `tcl-8.0.3-20.i386.rpm`, a newer one like `tcl-8.0.5-30.i386.rpm` for RedHat 6.1 should also do.

Apart from relinking, the installation is straightforward:

```
# su - oraids
# export TERM=xterm
# export ORACLE_TERM=xterm
# export ORACLE_HOME=/oracle/IDS
# cd $ORACLE_HOME/orainst_sap
# ./orainst
```

Confirm all screens with **Enter** until the software is installed, except that one has to deselect the *Oracle On-Line Text Viewer*, as this is not currently available for Linux. **Oracle** then wants to relink with `i386-glibc20-linux-gcc` instead of the available `gcc`, `egcs` or `i386-redhat-linux-gcc`.

Due to time constraints we decided to use the binaries from an **Oracle 8.0.5 PreProduction** release, after the first attempt at getting the version from the RDBMS CD working, failed, and finding and accessing the correct RPMs was a nightmare at that time.

10.7.9.2. Installing the Oracle 8.0.5 Pre-production Release for Linux (Kernel 2.0.33)

This installation is quite easy. Mount the CD, start the installer. It will then ask for the location of the Oracle home directory, and copy all binaries there. We did not delete the remains of our previous RDBMS installation tries, though.

Afterwards, **Oracle** Database could be started with no problems.

10.7.10. Installing the Oracle 8.1.7 Linux Tarball

Take the tarball `oracle81732.tgz` you produced from the installation directory on a Linux system and untar it to `/oracle/SID/817_32/`.

10.7.11. Continue with SAP R/3 Installation

First check the environment settings of users `idsamd` (`sidadm`) and `oraids` (`orasid`). They should now both have the files `.profile`, `.login` and `.cshrc` which are all using `hostname`. In case the system's hostname is the fully qualified name, you need to change `hostname` to `hostname -s` within all three files.

10.7.11.1. Database Load

Afterwards, `R3SETUP` can either be restarted or continued (depending on whether exit was chosen or not). `R3SETUP` then creates the tablespaces and loads the data (for 46B IDES, from `EXPORT1` to `EXPORT6`, for 46C from `DISK1` to `DISK4`) with `R3load` into the database.

When the database load is finished (might take a few hours), some passwords are requested. For test installations, one can use the well known default passwords (use different ones if security is an issue!):

Question	Input
Enter Password for <code>sapr3</code>	<code>sap</code> Enter
Confirm Password for <code>sapr3</code>	<code>sap</code> Enter
Enter Password for <code>sys</code>	<code>change_on_install</code> Enter
Confirm Password for <code>sys</code>	<code>change_on_install</code> Enter
Enter Password for <code>system</code>	<code>manager</code> Enter
Confirm Password for <code>system</code>	<code>manager</code> Enter

At this point We had a few problems with `dipgntab` during the 4.6B installation.

10.7.11.2. Listener

Start the **Oracle** Listener as user `orasid` as follows:

```
% umask 0; lsnrctl start
```

Otherwise you might get the error `ORA-12546` as the sockets will not have the correct permissions. See SAP Note 072984.

10.7.11.3. Updating MNLS Tables

If you plan to import non-Latin-1 languages into the **SAP** system, you have to update the Multi National Language Support tables. This is described in the SAP OSS Notes 15023 and 45619. Otherwise, you can skip this question during **SAP** installation.

Notatka: If you do not need MNLS, it is still necessary to check the table `TCPDB` and initializing it if this has not been done. See SAP note 0015023 and 0045619 for further information.

10.7.12. Post-installation Steps

10.7.12.1. Request SAP R/3 License Key

You have to request your **SAP R/3** License Key. This is needed, as the temporary license that was installed during installation is only valid for four weeks. First get the hardware key. Log on as user `idsadm` and call `saplicense`:

```
# /sapmnt/IDS/exe/saplicense -get
```

Calling `saplicense` without parameters gives a list of options. Upon receiving the license key, it can be installed using:

```
# /sapmnt/IDS/exe/saplicense -install
```

You are then required to enter the following values:

```
SAP SYSTEM ID    = SID, 3 chars
CUSTOMER KEY     = hardware key, 11 chars
INSTALLATION NO = installation, 10 digits
EXPIRATION DATE  = yyyyymmdd, usually "99991231"
LICENSE KEY      = license key, 24 chars
```

10.7.12.2. Creating Users

Create a user within client 000 (for some tasks required to be done within client 000, but with a user different from users `sap*` and `ddic`). As a user name, We usually choose `wartung` (or `service` in English). Profiles required are `sap_new` and `sap_all`. For additional safety the passwords of default users within all clients should be changed (this includes users `sap*` and `ddic`).

10.7.12.3. Configure Transport System, Profile, Operation Modes, Etc.

Within client 000, user different from `ddic` and `sap*`, do at least the following:

Task	Transaction
Configure Transport System, e.g. as <i>Stand-Alone Transport Domain Entity</i>	STMS
Create / Edit Profile for System	RZ10
Maintain Operation Modes and Instances	RZ04

These and all the other post-installation steps are thoroughly described in **SAP** installation guides.

10.7.12.4. Edit `initSID.sap` (`initIDS.sap`)

The file `/oracle/IDS/dbs/initIDS.sap` contains the **SAP** backup profile. Here the size of the tape to be used,

type of compression and so on need to be defined. To get this running with `sapdba / brbackup`, we changed the following values:

```
compress = hardware
archive_function = copy_delete_save
cpio_flags = "-ov --format=newc --block-size=128 --quiet"
cpio_in_flags = "-iuv --block-size=128 --quiet"
tape_size = 38000M
tape_address = /dev/nsa0
tape_address_rew = /dev/sa0
```

Explanations:

`compress`: The tape we use is a HP DLT1 which does hardware compression.

`archive_function`: This defines the default behavior for saving Oracle archive logs: new logfiles are saved to tape, already saved logfiles are saved again and are then deleted. This prevents lots of trouble if you need to recover the database, and one of the archive-tapes has gone bad.

`cpio_flags`: Default is to use `-B` which sets block size to 5120 Bytes. For DLT Tapes, HP recommends at least 32 K block size, so we used `--block-size=128` for 64 K. `--format=newc` is needed because we have inode numbers greater than 65535. The last option `--quiet` is needed as otherwise `brbackup` complains as soon as `cpio` outputs the numbers of blocks saved.

`cpio_in_flags`: Flags needed for loading data back from tape. Format is recognized automatically.

`tape_size`: This usually gives the raw storage capability of the tape. For security reason (we use hardware compression), the value is slightly lower than the actual value.

`tape_address`: The non-rewindable device to be used with `cpio`.

`tape_address_rew`: The rewindable device to be used with `cpio`.

10.7.12.5. Configuration Issues after Installation

The following **SAP** parameters should be tuned after installation (examples for IDES 46B, 1 GB memory):

Name	Value
ztta/roll_extension	250000000
abap/heap_area_dia	300000000
abap/heap_area_nondia	400000000
em/initial_size_MB	256
em/blocksize_kB	1024
ipc/shm_psize_40	70000000

SAP Note 0013026:

Name	Value
ztta/dynpro_area	2500000

SAP Note 0157246:

Name	Value
rdisp/ROLL_MAXFS	16000
rdisp/PG_MAXFS	30000

Notatka: With the above parameters, on a system with 1 gigabyte of memory, one may find memory consumption similar to:

Mem: 547M Active, 305M Inact, 109M Wired, 40M Cache, 112M Buf, 3492K Free

10.7.13. Problems during Installation

10.7.13.1. Restart R3SETUP after Fixing a Problem

R3SETUP stops if it encounters an error. If you have looked at the corresponding logfiles and fixed the error, you have to start R3SETUP again, usually selecting REPEAT as option for the last step R3SETUP complained about.

To restart R3SETUP, just start it with the corresponding R3S file:

```
# ./R3SETUP -f CENTRDB.R3S
```

for 4.6B, or with

```
# ./R3SETUP -f CENTRAL.R3S
```

for 4.6C, no matter whether the error occurred with CENTRAL.R3S or DATABASE.R3S.

Notatka: At some stages, R3SETUP assumes that both database and **SAP** processes are up and running (as those were steps it already completed). Should errors occur and for example the database could not be started, you have to start both database and **SAP** by hand after you fixed the errors and before starting R3SETUP again.

Do not forget to also start the **Oracle** listener again (as `orasisd` with `umask 0; lsnrctl start`) if it was also stopped (for example due to a necessary reboot of the system).

10.7.13.2. OSUSERSIDADM_IND_ORA during R3SETUP

If R3SETUP complains at this stage, edit the template file R3SETUP used at that time (CENTRDB.R3S (4.6B) or either CENTRAL.R3S or DATABASE.R3S (4.6C)). Locate [OSUSERSIDADM_IND_ORA] or search for the only STATUS=ERROR entry and edit the following values:

```
HOME=/home/sidadm (was empty)
STATUS=OK (had status ERROR)
```

Then you can restart R3SETUP again.

10.7.13.3. OSUSERDBSID_IND_ORA during R3SETUP

Possibly R3SETUP also complains at this stage. The error here is similar to the one in phase OSUSERSIDADM_IND_ORA. Just edit the template file R3SETUP used at that time (CENTRDB.R3S (4.6B) or either CENTRAL.R3S or DATABASE.R3S (4.6C)). Locate [OSUSERDBSID_IND_ORA] or search for the only STATUS=ERROR entry and edit the following value in that section:

```
STATUS=OK
```

Then restart R3SETUP.

10.7.13.4. oraview.vrf FILE NOT FOUND during Oracle Installation

You have not deselected *Oracle On-Line Text Viewer* before starting the installation. This is marked for installation even though this option is currently not available for Linux. Deselect this product inside the **Oracle** installation menu and restart installation.

10.7.13.5. TEXTENV_INVALID during R3SETUP, RFC or SAPgui Start

If this error is encountered, the correct locale is missing. SAP Note 0171356 lists the necessary RPMs that need be installed (e.g. saplocales-1.0-3, saposcheck-1.0-1 for RedHat 6.1). In case you ignored all the related errors and set the corresponding STATUS from ERROR to OK (in CENTRDB.R3S) every time R3SETUP complained and just restarted R3SETUP, the **SAP** system will not be properly configured and you will then not be able to connect to the system with a **SAPgui**, even though the system can be started. Trying to connect with the old Linux **SAPgui** gave the following messages:

```
Sat May 5 14:23:14 2001
*** ERROR => no valid userarea given [trgmsggo. 0401]
Sat May 5 14:23:22 2001
*** ERROR => ERROR NR 24 occurred [trgmsggi. 0410]
*** ERROR => Error when generating text environment. [trgmsggi. 0435]
*** ERROR => function failed [trgmsggi. 0447]
*** ERROR => no socket operation allowed [trxio.c 3363]
Speicherzugriffsfehler
```

This behavior is due to **SAP R/3** being unable to correctly assign a locale and also not being properly configured itself (missing entries in some database tables). To be able to connect to **SAP**, add the following entries to file DEFAULT.PFL (see Note 0043288):

```
abap/set_etct_env_at_new_mode = 0
install/collate/active = 0
rscp/TCP0B = TCP0B
```

Restart the **SAP** system. Now you can connect to the system, even though country-specific language settings might not work as expected. After correcting country settings (and providing the correct locales), these entries can be removed from DEFAULT.PFL and the **SAP** system can be restarted.

10.7.13.6. ORA-00001

This error only happened with **Oracle 8.1.7** on FreeBSD. The reason was that the **Oracle** database could not initialize itself properly and crashed, leaving semaphores and shared memory on the system. The next try to start the database then returned ORA-00001.

Find them with `ipcs -a` and remove them with `ipcrm`.

10.7.13.7. ORA-00445 (Background Process PMON Did Not Start)

This error happened with **Oracle 8.1.7**. This error is reported if the database is started with the usual `startsap` script (for example `startsap_majestix_00`) as user `prdadm`.

A possible workaround is to start the database as user `oraprd` instead with `svrmgrl`:

```
% svrmgrl
SVRMGR> connect internal;
SVRMGR> startup;
SVRMGR> exit
```

10.7.13.8. ORA-12546 (Start Listener with Correct Permissions)

Start the **Oracle** listener as user `oraids` with the following commands:

```
# umask 0; lsnrctl start
```

Otherwise you might get ORA-12546 as the sockets will not have the correct permissions. See SAP Note 0072984.

10.7.13.9. ORA-27102 (Out of Memory)

This error happened whilst trying to use values for `MAXDSIZ` and `DFLDSIZ` greater than 1 GB (1024x1024x1024). Additionally, we got Linux Error 12: Cannot allocate memory.

10.7.13.10. [DIPGNTAB_IND_IND] during R3SETUP

In general, see SAP Note 0130581 (R3SETUP step DIPGNTAB terminates). During the IDES-specific installation, for some reason the installation process was not using the proper **SAP** system name „IDS”, but the empty string " " instead. This leads to some minor problems with accessing directories, as the paths are generated dynamically using `SID` (in this case IDS). So instead of accessing:

```
/usr/sap/IDS/SYS/...
/usr/sap/IDS/DVMGS00
```

the following paths were used:

```
/usr/sap//SYS/...
/usr/sap/D00
```

To continue with the installation, we created a link and an additional directory:

```
# pwd
```

```
/compat/linux/usr/sap
# ls -l
total 4
drwxr-xr-x 3  idsadm sapsys 512 May 5 11:20 D00
drwxr-x--x 5  idsadm sapsys 512 May 5 11:35 IDS
lrwxr-xr-x 1  root   sapsys 7 May 5 11:35 SYS -> IDS/SYS
drwxrwxr-x 2  idsadm sapsys 512 May 5 13:00 tmp
drwxrwxr-x 11 idsadm sapsys 512 May 4 14:20 trans
```

We also found SAP Notes (0029227 and 0008401) describing this behavior. We did not encounter any of these problems with the **SAP 4.6C** installation.

10.7.13.11. [RFCRSWBOINI_IND_IND] during R3SETUP

During installation of **SAP 4.6C**, this error was just the result of another error happening earlier during installation. In this case, you have to look through the corresponding logfiles and correct the real problem.

If after looking through the logfiles this error is indeed the correct one (check the SAP Notes), you can set STATUS of the offending step from ERROR to OK (file CENTRDB.R3S) and restart R3SETUP. After installation, you have to execute the report RSWBOINS from transaction SE38. See SAP Note 0162266 for additional information about phase RFCRSWBOINI and RFCRADDBDIF.

10.7.13.12. [RFCRADDBDIF_IND_IND] during R3SETUP

Here the same restrictions apply: make sure by looking through the logfiles, that this error is not caused by some previous problems.

If you can confirm that SAP Note 0162266 applies, just set STATUS of the offending step from ERROR to OK (file CENTRDB.R3S) and restart R3SETUP. After installation, you have to execute the report RADDBDIF from transaction SE38.

10.7.13.13. sigaction sig31: File size limit exceeded

This error occurred during start of **SAP** processes *disp+work*. If starting **SAP** with the `startsap` script, subprocesses are then started which detach and do the dirty work of starting all other **SAP** processes. As a result, the script itself will not notice if something goes wrong.

To check whether the **SAP** processes did start properly, have a look at the process status with `ps ax | grep SID`, which will give you a list of all **Oracle** and **SAP** processes. If it looks like some processes are missing or if you cannot connect to the **SAP** system, look at the corresponding logfiles which can be found at `/usr/sap/SID/DVEBMGSnr/work/`. The files to look at are `dev_ms` and `dev_disp`.

Signal 31 happens here if the amount of shared memory used by **Oracle** and **SAP** exceed the one defined within the kernel configuration file and could be resolved by using a larger value:

```
# larger value for 46C production systems:
options SHMMAXPGS=393216
# smaller value sufficient for 46B:
#options SHMMAXPGS=262144
```

10.7.13.14. Start of `saposcol` Failed

There are some problems with the program `saposcol` (version 4.6D). The **SAP** system is using `saposcol` to collect data about the system performance. This program is not needed to use the **SAP** system, so this problem can be considered a minor one. The older versions (4.6B) does work, but does not collect all the data (many calls will just return 0, for example for CPU usage).

10.8. Advanced Topics

If you are curious as to how the Linux binary compatibility works, this is the section you want to read. Most of what follows is based heavily on an email written to FreeBSD chat mailing list (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat>) by Terry Lambert <tlambert@primenet.com> (Message ID: <199906020108.SAA07001@usr09.primenet.com>).

10.8.1. How Does It Work?

FreeBSD has an abstraction called an „execution class loader”. This is a wedge into the `execve(2)` system call.

What happens is that FreeBSD has a list of loaders, instead of a single loader with a fallback to the `#!` loader for running any shell interpreters or shell scripts.

Historically, the only loader on the UNIX platform examined the magic number (generally the first 4 or 8 bytes of the file) to see if it was a binary known to the system, and if so, invoked the binary loader.

If it was not the binary type for the system, the `execve(2)` call returned a failure, and the shell attempted to start executing it as shell commands.

The assumption was a default of „whatever the current shell is”.

Later, a hack was made for `sh(1)` to examine the first two characters, and if they were `:\n`, then it invoked the `csh(1)` shell instead (we believe SCO first made this hack).

What FreeBSD does now is go through a list of loaders, with a generic `#!` loader that knows about interpreters as the characters which follow to the next whitespace next to last, followed by a fallback to `/bin/sh`.

For the Linux ABI support, FreeBSD sees the magic number as an ELF binary (it makes no distinction between FreeBSD, Solaris, Linux, or any other OS which has an ELF image type, at this point).

The ELF loader looks for a specialized *brand*, which is a comment section in the ELF image, and which is not present on SVR4/Solaris ELF binaries.

For Linux binaries to function, they must be *branded* as type `Linux` from `brandelf(1)`:

```
# brandelf -t Linux file
```

When this is done, the ELF loader will see the `Linux` brand on the file.

When the ELF loader sees the `Linux` brand, the loader replaces a pointer in the `proc` structure. All system calls are indexed through this pointer (in a traditional UNIX system, this would be the `sysent[]` structure array, containing the system calls). In addition, the process is flagged for special handling of the trap vector for the signal trampoline code, and several other (minor) fix-ups that are handled by the Linux kernel module.

The Linux system call vector contains, among other things, a list of `sysent[]` entries whose addresses reside in the kernel module.

When a system call is called by the Linux binary, the trap code dereferences the system call function pointer off the `proc` structure, and gets the Linux, not the FreeBSD, system call entry points.

In addition, the Linux mode dynamically *reroots* lookups; this is, in effect, what the `union` option to file system mounts (*not* the `unionfs` file system type!) does. First, an attempt is made to lookup the file in the `/compat/linux/original-path` directory, *then* only if that fails, the lookup is done in the `/original-path` directory. This makes sure that binaries that require other binaries can run (e.g., the Linux toolchain can all run under Linux ABI support). It also means that the Linux binaries can load and execute FreeBSD binaries, if there are no corresponding Linux binaries present, and that you could place a `uname(1)` command in the `/compat/linux` directory tree to ensure that the Linux binaries could not tell they were not running on Linux.

In effect, there is a Linux kernel in the FreeBSD kernel; the various underlying functions that implement all of the services provided by the kernel are identical to both the FreeBSD system call table entries, and the Linux system call table entries: file system operations, virtual memory operations, signal delivery, System V IPC, etc... The only difference is that FreeBSD binaries get the FreeBSD *glue* functions, and Linux binaries get the Linux *glue* functions (most older OS's only had their own *glue* functions: addresses of functions in a static global `sysent[]` structure array, instead of addresses of functions dereferenced off a dynamically initialized pointer in the `proc` structure of the process making the call).

Which one is the native FreeBSD ABI? It does not matter. Basically the only difference is that (currently; this could easily be changed in a future release, and probably will be after this) the FreeBSD *glue* functions are statically linked into the kernel, and the Linux *glue* functions can be statically linked, or they can be accessed via a kernel module.

Yeah, but is this really emulation? No. It is an ABI implementation, not an emulation. There is no emulator (or simulator, to cut off the next question) involved.

So why is it sometimes called „Linux emulation”? To make it hard to sell FreeBSD! Really, it is because the historical implementation was done at a time when there was really no word other than that to describe what was going on; saying that FreeBSD ran Linux binaries was not true, if you did not compile the code in or load a module, and there needed to be a word to describe what was being loaded—hence „the Linux emulator”.

III. Administracja systemem

Pozostałe rozdziały Podręcznika omawiają wszystkie aspekty administracji systemem FreeBSD. Każdy z nich rozpoczyna się on wyjaśnieniami czego nauczymy się przeczytawszy dany rozdział, a także co powinniśmy wiedzieć przed przystąpieniem do jego lektury.

Rozdziały zostały tak napisane, by móc sięgnąć po nie gdy potrzebujemy danych informacji. Nie ma przymusu czytania ich w żadnej określonej kolejności, ani też przeczytania wszystkich przed rozpoczęciem pracy z FreeBSD.

Rozdział 11. Configuration and Tuning

Written by Chern Lee. Based on a tutorial written by Mike Smith. Also based on tuning(7) written by Matt Dillon.

11.1. Synopsis

One of the important aspects of FreeBSD is system configuration. Correct system configuration will help prevent headaches during future upgrades. This chapter will explain much of the FreeBSD configuration process, including some of the parameters which can be set to tune a FreeBSD system.

After reading this chapter, you will know:

- How to efficiently work with file systems and swap partitions.
- The basics of `rc.conf` configuration and `/usr/local/etc/rc.d` startup systems.
- How to configure and test a network card.
- How to configure virtual hosts on your network devices.
- How to use the various configuration files in `/etc`.
- How to tune FreeBSD using `sysctl` variables.
- How to tune disk performance and modify kernel limitations.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Rozdział 3).
- Be familiar with the basics of kernel configuration/compilation (Rozdział 8).

11.2. Initial Configuration

11.2.1. Partition Layout

11.2.1.1. Base Partitions

When laying out file systems with `bsdlabel(8)` or `sysinstall(8)`, remember that hard drives transfer data faster from the outer tracks to the inner. Thus smaller and heavier-accessed file systems should be closer to the outside of the drive, while larger partitions like `/usr` should be placed toward the inner. It is a good idea to create partitions in a similar order to: `root`, `swap`, `/var`, `/usr`.

The size of `/var` reflects the intended machine usage. `/var` is used to hold mailboxes, log files, and printer spools. Mailboxes and log files can grow to unexpected sizes depending on how many users exist and how long log files are kept. Most users would never require a gigabyte, but remember that `/var/tmp` must be large enough to contain packages.

The `/usr` partition holds much of the files required to support the system, the ports(7) collection (recommended) and the source code (optional). Both of which are optional at install time. At least 2 gigabytes would be recommended for this partition.

When selecting partition sizes, keep the space requirements in mind. Running out of space in one partition while barely using another can be a hassle.

Notatka: Some users have found that `sysinstall(8)`'s `Auto-defaults` partition sizer will sometimes select smaller than adequate `/var` and `/` partitions. Partition wisely and generously.

11.2.1.2. Swap Partition

As a rule of thumb, the swap partition should be about double the size of system memory (RAM). For example, if the machine has 128 megabytes of memory, the swap file should be 256 megabytes. Systems with less memory may perform better with more swap. Less than 256 megabytes of swap is not recommended and memory expansion should be considered. The kernel's VM paging algorithms are tuned to perform best when the swap partition is at least two times the size of main memory. Configuring too little swap can lead to inefficiencies in the VM page scanning code and might create issues later if more memory is added.

On larger systems with multiple SCSI disks (or multiple IDE disks operating on different controllers), it is recommend that a swap is configured on each drive (up to four drives). The swap partitions should be approximately the same size. The kernel can handle arbitrary sizes but internal data structures scale to 4 times the largest swap partition. Keeping the swap partitions near the same size will allow the kernel to optimally stripe swap space across disks. Large swap sizes are fine, even if swap is not used much. It might be easier to recover from a runaway program before being forced to reboot.

11.2.1.3. Why Partition?

Several users think a single large partition will be fine, but there are several reasons why this is a bad idea. First, each partition has different operational characteristics and separating them allows the file system to tune accordingly. For example, the root and `/usr` partitions are read-mostly, without much writing. While a lot of reading and writing could occur in `/var` and `/var/tmp`.

By properly partitioning a system, fragmentation introduced in the smaller write heavy partitions will not bleed over into the mostly-read partitions. Keeping the write-loaded partitions closer to the disk's edge, will increase I/O performance in the partitions where it occurs the most. Now while I/O performance in the larger partitions may be needed, shifting them more toward the edge of the disk will not lead to a significant performance improvement over moving `/var` to the edge. Finally, there are safety concerns. A smaller, neater root partition which is mostly read-only has a greater chance of surviving a bad crash.

11.3. Core Configuration

The principal location for system configuration information is within `/etc/rc.conf`. This file contains a wide range of configuration information, principally used at system startup to configure the system. Its name directly implies this; it is configuration information for the `rc*` files.

An administrator should make entries in the `rc.conf` file to override the default settings from `/etc/defaults/rc.conf`. The defaults file should not be copied verbatim to `/etc` - it contains default values, not examples. All system-specific changes should be made in the `rc.conf` file itself.

A number of strategies may be applied in clustered applications to separate site-wide configuration from system-specific configuration in order to keep administration overhead down. The recommended approach is to place site-wide configuration into another file, such as `/etc/rc.conf.site`, and then include this file into `/etc/rc.conf`, which will contain only system-specific information.

As `rc.conf` is read by `sh(1)` it is trivial to achieve this. For example:

- `rc.conf`:

```
. /etc/rc.conf.site
hostname="node15.example.com"
network_interfaces="fxp0 lo0"
ifconfig_fxp0="inet 10.1.1.1"
```

- `rc.conf.site`:

```
defaultrouter="10.1.1.254"
saver="daemon"
blanktime="100"
```

The `rc.conf.site` file can then be distributed to every system using `rsync` or a similar program, while the `rc.conf` file remains unique.

Upgrading the system using `sysinstall(8)` or `make world` will not overwrite the `rc.conf` file, so system configuration information will not be lost.

11.4. Application Configuration

Typically, installed applications have their own configuration files, with their own syntax, etc. It is important that these files be kept separate from the base system, so that they may be easily located and managed by the package management tools.

Typically, these files are installed in `/usr/local/etc`. In the case where an application has a large number of configuration files, a subdirectory will be created to hold them.

Normally, when a port or package is installed, sample configuration files are also installed. These are usually identified with a `.default` suffix. If there are no existing configuration files for the application, they will be created by copying the `.default` files.

For example, consider the contents of the directory `/usr/local/etc/apache`:

```
-rw-r--r--  1 root  wheel   2184 May 20   1998 access.conf
-rw-r--r--  1 root  wheel   2184 May 20   1998 access.conf.default
-rw-r--r--  1 root  wheel   9555 May 20   1998 httpd.conf
-rw-r--r--  1 root  wheel   9555 May 20   1998 httpd.conf.default
-rw-r--r--  1 root  wheel   12205 May 20   1998 magic
-rw-r--r--  1 root  wheel   12205 May 20   1998 magic.default
-rw-r--r--  1 root  wheel    2700 May 20   1998 mime.types
-rw-r--r--  1 root  wheel    2700 May 20   1998 mime.types.default
-rw-r--r--  1 root  wheel    7980 May 20   1998 srm.conf
-rw-r--r--  1 root  wheel    7933 May 20   1998 srm.conf.default
```

The file sizes show that only the `srm.conf` file has been changed. A later update of the **Apache** port would not overwrite this changed file.

11.5. Starting Services

Contributed by Tom Rhodes.

Many users choose to install third party software on FreeBSD from the Ports Collection. In many of these situations it may be necessary to configure the software in a manner which will allow it to be started upon system initialization. Services, such as `mail/postfix` or `www/apache13` are just two of the many software packages which may be started during system initialization. This section explains the procedures available for starting third party software.

In FreeBSD, most included services, such as `cron(8)`, are started through the system start up scripts. These scripts may differ depending on FreeBSD or vendor version; however, the most important aspect to consider is that their start up configuration can be handled through simple startup scripts.

Before the advent of `rc.d`, applications would drop a simple start up script into the `/usr/local/etc/rc.d` directory which would be read by the system initialization scripts. These scripts would then be executed during the latter stages of system start up.

While many individuals have spent hours trying to merge the old configuration style into the new system, the fact remains that some third party utilities still require a script simply dropped into the aforementioned directory. The subtle differences in the scripts depend whether or not `rc.d` is being used. Prior to FreeBSD 5.1 the old configuration style is used and in almost all cases a new style script would do just fine.

While every script must meet some minimal requirements, most of the time these requirements are FreeBSD version agnostic. Each script must have a `.sh` extension appended to the end and every script must be executable by the system. The latter may be achieved by using the `chmod` command and setting the unique permissions of 755. There should also be, at minimal, an option to `start` the application and an option to `stop` the application.

The simplest start up script would probably look a little bit like this one:

```
#!/bin/sh
echo -n ' utility'

case "$1" in
start)
    /usr/local/bin/utility
    ;;
stop)
    kill -9 `cat /var/run/utility.pid`
    ;;
*)
    echo "Usage: `basename $0` {start|stop}" >&2
    exit 64
    ;;
esac

exit 0
```

This script provides for a `stop` and `start` option for the application hereto referred simply as `utility`.

Could be started manually with:

```
# /usr/local/etc/rc.d/utility.sh start
```

While not all third party software requires the line in `rc.conf`, almost every day a new port will be modified to accept this configuration. Check the final output of the installation for more information on a specific application. Some third party software will provide start up scripts which permit the application to be used with `rc.d`; although, this will be discussed in the next section.

11.5.1. Extended Application Configuration

Now that FreeBSD includes `rc.d`, configuration of application startup has become easier, and more featureful. Using the key words discussed in the `rc.d` section, applications may now be set to start after certain other services for example DNS; may permit extra flags to be passed through `rc.conf` in place of hard coded flags in the start up script, etc. A basic script may look similar to the following:

```
#!/bin/sh
#
# PROVIDE: utility
# REQUIRE: DAEMON
# KEYWORD: shutdown

. /etc/rc.subr

name=utility
rcvar=utility_enable

command="/usr/local/sbin/utility"

load_rc_config $name

#
# DO NOT CHANGE THESE DEFAULT VALUES HERE
# SET THEM IN THE /etc/rc.conf FILE
#
utility_enable=${utility_enable-"NO"}
pidfile=${utility_pidfile-"/var/run/utility.pid"}

run_rc_command "$1"
```

This script will ensure that the provided **utility** will be started after the `daemon` service. It also provides a method for setting and tracking the PID, or process ID file.

This application could then have the following line placed in `/etc/rc.conf`:

```
utility_enable="YES"
```

This new method also allows for easier manipulation of the command line arguments, inclusion of the default functions provided in `/etc/rc.subr`, compatibility with the `rcorder(8)` utility and provides for easier configuration via the `rc.conf` file.

11.5.2. Using Services to Start Services

Other services, such as POP3 server daemons, IMAP, etc. could be started using the `inetd(8)`. This involves installing the service utility from the Ports Collection with a configuration line appended to the `/etc/inetd.conf` file, or uncommenting one of the current configuration lines. Working with **inetd** and its configuration is described in depth in the `inetd` section.

In some cases, it may be more plausible to use the `cron(8)` daemon to start system services. This approach has a number of advantages because `cron` runs these processes as the `crontab`'s file owner. This allows regular users to start and maintain some applications.

The `cron` utility provides a unique feature, `@reboot`, which may be used in place of the time specification. This will cause the job to be run when `cron(8)` is started, normally during system initialization.

11.6. Configuring the `cron` Utility

Contributed by Tom Rhodes.

One of the most useful utilities in FreeBSD is `cron(8)`. The `cron` utility runs in the background and constantly checks the `/etc/crontab` file. The `cron` utility also checks the `/var/cron/tabs` directory, in search of new `crontab` files. These `crontab` files store information about specific functions which `cron` is supposed to perform at certain times.

The `cron` utility uses two different types of configuration files, the system `crontab` and user `crontabs`. The only difference between these two formats is the sixth field. In the system `crontab`, the sixth field is the name of a user for the command to run as. This gives the system `crontab` the ability to run commands as any user. In a user `crontab`, the sixth field is the command to run, and all commands run as the user who created the `crontab`; this is an important security feature.

Notatka: User `crontabs` allow individual users to schedule tasks without the need for `root` privileges. Commands in a user's `crontab` run with the permissions of the user who owns the `crontab`.

The `root` user can have a user `crontab` just like any other user. This one is different from `/etc/crontab` (the system `crontab`). Because of the system `crontab`, there is usually no need to create a user `crontab` for `root`.

Let us take a look at the `/etc/crontab` file (the system `crontab`):

```
# /etc/crontab - root's crontab for FreeBSD
#
# $FreeBSD: src/etc/crontab,v 1.32 2002/11/22 16:13:39 tom Exp $
# ❶
#
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin ❷
HOME=/var/log
#
#
#minute hour      mday      month      wday      who        command ❸
#
#
```

```
* /5      *      *      *      *      root      /usr/libexec/atrun ④
```

- ❶ Like most FreeBSD configuration files, the # character represents a comment. A comment can be placed in the file as a reminder of what and why a desired action is performed. Comments cannot be on the same line as a command or else they will be interpreted as part of the command; they must be on a new line. Blank lines are ignored.
- ❷ First, the environment must be defined. The equals (=) character is used to define any environment settings, as with this example where it is used for the SHELL, PATH, and HOME options. If the shell line is omitted, cron will use the default, which is sh. If the PATH variable is omitted, no default will be used and file locations will need to be absolute. If HOME is omitted, cron will use the invoking users home directory.
- ❸ This line defines a total of seven fields. Listed here are the values minute, hour, mday, month, wday, who, and command. These are almost all self explanatory. minute is the time in minutes the command will be run. hour is similar to the minute option, just in hours. mday stands for day of the month. month is similar to hour and minute, as it designates the month. The wday option stands for day of the week. All these fields must be numeric values, and follow the twenty-four hour clock. The who field is special, and only exists in the /etc/crontab file. This field specifies which user the command should be run as. When a user installs his or her crontab file, they will not have this option. Finally, the command option is listed. This is the last field, so naturally it should designate the command to be executed.
- ❹ This last line will define the values discussed above. Notice here we have a */5 listing, followed by several more * characters. These * characters mean „first-last”, and can be interpreted as every time. So, judging by this line, it is apparent that the atrun command is to be invoked by root every five minutes regardless of what day or month it is. For more information on the atrun command, see the atrun(8) manual page.

Commands can have any number of flags passed to them; however, commands which extend to multiple lines need to be broken with the backslash „\” continuation character.

This is the basic set up for every crontab file, although there is one thing different about this one. Field number six, where we specified the username, only exists in the system /etc/crontab file. This field should be omitted for individual user crontab files.

11.6.1. Installing a Crontab

WAŻNE: You must not use the procedure described here to edit/install the system crontab. Simply use your favorite editor: the cron utility will notice that the file has changed and immediately begin using the updated version. See this FAQ entry (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/books/faq/admin.html#ROOT-NOT-FOUND-CRON-ERRORS) for more information.

To install a freshly written user crontab, first use your favorite editor to create a file in the proper format, and then use the crontab utility. The most common usage is:

```
% crontab crontab-file
```

In this example, crontab-file is the filename of a crontab that was previously created.

There is also an option to list installed crontab files: just pass the -l option to crontab and look over the output.

For users who wish to begin their own crontab file from scratch, without the use of a template, the `crontab -e` option is available. This will invoke the selected editor with an empty file. When the file is saved, it will be automatically installed by the `crontab` command.

If you later want to remove your user crontab completely, use `crontab` with the `-r` option.

11.7. Using rc under FreeBSD

Contributed by Tom Rhodes.

In 2002 FreeBSD integrated the NetBSD `rc.d` system for system initialization. Users should notice the files listed in the `/etc/rc.d` directory. Many of these files are for basic services which can be controlled with the `start`, `stop`, and `restart` options. For instance, `sshd(8)` can be restarted with the following command:

```
# /etc/rc.d/sshd restart
```

This procedure is similar for other services. Of course, services are usually started automatically at boot time as specified in `rc.conf(5)`. For example, enabling the Network Address Translation daemon at startup is as simple as adding the following line to `/etc/rc.conf`:

```
natd_enable="YES"
```

If a `natd_enable="NO"` line is already present, then simply change the `NO` to `YES`. The `rc` scripts will automatically load any other dependent services during the next reboot, as described below.

Since the `rc.d` system is primarily intended to start/stop services at system startup/shutdown time, the standard `start`, `stop` and `restart` options will only perform their action if the appropriate `/etc/rc.conf` variables are set. For instance the above `sshd restart` command will only work if `sshd_enable` is set to `YES` in `/etc/rc.conf`. To start, stop or restart a service regardless of the settings in `/etc/rc.conf`, the commands should be prefixed with „force”. For instance to restart `sshd` regardless of the current `/etc/rc.conf` setting, execute the following command:

```
# /etc/rc.d/sshd forcerestart
```

It is easy to check if a service is enabled in `/etc/rc.conf` by running the appropriate `rc.d` script with the option `rcvar`. Thus, an administrator can check that `sshd` is in fact enabled in `/etc/rc.conf` by running:

```
# /etc/rc.d/sshd rcvar
# sshd
$sshd_enable=YES
```

Notatka: The second line (`# sshd`) is the output from the `sshd` command, not a `root` console.

To determine if a service is running, a `status` option is available. For instance to verify that `sshd` is actually started:

```
# /etc/rc.d/sshd status
sshd is running as pid 433.
```

In some cases it is also possible to `reload` a service. This will attempt to send a signal to an individual service, forcing the service to reload its configuration files. In most cases this means sending the service a `SIGHUP` signal. Support for this feature is not included for every service.

The `rc.d` system is not only used for network services, it also contributes to most of the system initialization. For instance, consider the `bgfsck` file. When this script is executed, it will print out the following message:

```
Starting background file system checks in 60 seconds.
```

Therefore this file is used for background file system checks, which are done only during system initialization.

Many system services depend on other services to function properly. For example, NIS and other RPC-based services may fail to start until after the `rpcbind` (portmapper) service has started. To resolve this issue, information about dependencies and other meta-data is included in the comments at the top of each startup script. The `rcorder(8)` program is then used to parse these comments during system initialization to determine the order in which system services should be invoked to satisfy the dependencies. The following words may be included at the top of each startup file:

- `PROVIDE`: Specifies the services this file provides.
- `REQUIRE`: Lists services which are required for this service. This file will run *after* the specified services.
- `BEFORE`: Lists services which depend on this service. This file will run *before* the specified services.

By using this method, an administrator can easily control system services without the hassle of „runlevels” like some other UNIX operating systems.

Additional information about the `rc.d` system can be found in the `rc(8)` and `rc.subr(8)` manual pages.

11.8. Setting Up Network Interface Cards

Contributed by Marc Fonvieille.

Nowadays we can not think about a computer without thinking about a network connection. Adding and configuring a network card is a common task for any FreeBSD administrator.

11.8.1. Locating the Correct Driver

Before you begin, you should know the model of the card you have, the chip it uses, and whether it is a PCI or ISA card. FreeBSD supports a wide variety of both PCI and ISA cards. Check the Hardware Compatibility List for your release to see if your card is supported.

Once you are sure your card is supported, you need to determine the proper driver for the card.

`/usr/src/sys/conf/NOTES` and `/usr/src/sys/arch/conf/NOTES` will give you the list of network interface drivers with some information about the supported chipsets/cards. If you have doubts about which driver is the correct one, read the manual page of the driver. The manual page will give you more information about the supported hardware and even the possible problems that could occur.

If you own a common card, most of the time you will not have to look very hard for a driver. Drivers for common network cards are present in the `GENERIC` kernel, so your card should show up during boot, like so:

```
dc0: <82c169 PNIC 10/100BaseTX> port 0xa000-0xa0ff mem 0xd3800000-0xd38000ff irq 15 at device 11.0 on pci0
```

```
dc0: Ethernet address: 00:a0:cc:da:da:da
miibus0: <MII bus> on dc0
ukphy0: <Generic IEEE 802.3u media interface> on miibus0
ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc1: <82c169 PNIC 10/100BaseTX> port 0x9800-0x98ff mem 0xd3000000-0xd30
000ff irq 11 at device 12.0 on pci0
dc1: Ethernet address: 00:a0:cc:da:da:db
miibus1: <MII bus> on dc1
ukphy1: <Generic IEEE 802.3u media interface> on miibus1
ukphy1: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
```

In this example, we see that two cards using the dc(4) driver are present on the system.

If the driver for your NIC is not present in `GENERIC`, you will need to load the proper driver to use your NIC. This may be accomplished in one of two ways:

- The easiest way is to simply load a kernel module for your network card with `kldload(8)`, or automatically at boot time by adding the appropriate line to the file `/boot/loader.conf`. Not all NIC drivers are available as modules; notable examples of devices for which modules do not exist are ISA cards.
- Alternatively, you may statically compile the support for your card into your kernel. Check `/usr/src/sys/conf/NOTES`, `/usr/src/sys/arch/conf/NOTES` and the manual page of the driver to know what to add in your kernel configuration file. For more information about recompiling your kernel, please see Rozdział 8. If your card was detected at boot by your kernel (`GENERIC`) you do not have to build a new kernel.

11.8.1.1. Using Windows NDIS Drivers

Unfortunately, there are still many vendors that do not provide schematics for their drivers to the open source community because they regard such information as trade secrets. Consequently, the developers of FreeBSD and other operating systems are left two choices: develop the drivers by a long and pain-staking process of reverse engineering or using the existing driver binaries available for the Microsoft Windows platforms. Most developers, including those involved with FreeBSD, have taken the latter approach.

Thanks to the contributions of Bill Paul (`wpaul`), as of FreeBSD 5.3-RELEASE there is „native” support for the Network Driver Interface Specification (NDIS). The FreeBSD NDISulator (otherwise known as Project Evil) takes a Windows driver binary and basically tricks it into thinking it is running on Windows. Because the `ndis(4)` driver is using a Windows binary, it is only usable on i386 and amd64 systems.

Notatka: The `ndis(4)` driver is designed to support mainly PCI, CardBus and PCMCIA devices, USB devices are not yet supported.

In order to use the NDISulator, you need three things:

1. Kernel sources
2. Windows XP driver binary (`.SYS` extension)
3. Windows XP driver configuration file (`.INF` extension)

Locate the files for your specific card. Generally, they can be found on the included CDs or at the vendors' websites. In the following examples, we will use `W32DRIVER.SYS` and `W32DRIVER.INF`.

Notatka: You can not use a Windows/i386 driver with FreeBSD/amd64, you must get a Windows/amd64 driver to make it work properly.

The next step is to compile the driver binary into a loadable kernel module. To accomplish this, as `root`, use `ndisgen(8)`:

```
# ndisgen /path/to/W32DRIVER.INF /path/to/W32DRIVER.SYS
```

The `ndisgen(8)` utility is interactive and will prompt for any extra information it requires; it will produce a kernel module in the current directory which can be loaded as follows:

```
# kldload ./W32DRIVER.ko
```

In addition to the generated kernel module, you must load the `ndis.ko` and `if_ndis.ko` modules. This should be automatically done when you load any module that depends on `ndis(4)`. If you want to load them manually, use the following commands:

```
# kldload ndis
# kldload if_ndis
```

The first command loads the NDIS miniport driver wrapper, the second loads the actual network interface.

Now, check `dmesg(8)` to see if there were any errors loading. If all went well, you should get output resembling the following:

```
ndis0: <Wireless-G PCI Adapter> mem 0xf4100000-0xf4101fff irq 3 at device 8.0 on pci1
ndis0: NDIS API version: 5.0
ndis0: Ethernet address: 0a:b1:2c:d3:4e:f5
ndis0: 11b rates: 1Mbps 2Mbps 5.5Mbps 11Mbps
ndis0: 11g rates: 6Mbps 9Mbps 12Mbps 18Mbps 36Mbps 48Mbps 54Mbps
```

From here you can treat the `ndis0` device like any other network interface (e.g., `dc0`).

You can configure the system to load the NDIS modules at boot time in the same way as with any other module.

First, copy the generated module, `W32DRIVER.ko`, to the `/boot/modules` directory. Then, add the following line to `/boot/loader.conf`:

```
W32DRIVER_load="YES"
```

11.8.2. Configuring the Network Card

Once the right driver is loaded for the network card, the card needs to be configured. As with many other things, the network card may have been configured at installation time by **sysinstall**.

To display the configuration for the network interfaces on your system, enter the following command:

```
% ifconfig
dc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.1.3 netmask 0xffffffff broadcast 192.168.1.255
    ether 00:a0:cc:da:da:da
    media: Ethernet autoselect (100baseTX <full-duplex>)
```

```

        status: active
dc1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 0xffffffff broadcast 10.0.0.255
    ether 00:a0:cc:da:da:db
    media: Ethernet 10baseT/UTP
    status: no carrier
lp0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    inet 127.0.0.1 netmask 0xff000000
tun0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500

```

Notatka: Old versions of FreeBSD may require the `-a` option following `ifconfig(8)`, for more details about the correct syntax of `ifconfig(8)`, please refer to the manual page. Note also that entries concerning IPv6 (`inet6` etc.) were omitted in this example.

In this example, the following devices were displayed:

- `dc0`: The first Ethernet interface
- `dc1`: The second Ethernet interface
- `lp0`: The parallel port interface
- `lo0`: The loopback device
- `tun0`: The tunnel device used by **ppp**

FreeBSD uses the driver name followed by the order in which one the card is detected at the kernel boot to name the network card. For example `sis2` would be the third network card on the system using the `sis(4)` driver.

In this example, the `dc0` device is up and running. The key indicators are:

1. UP means that the card is configured and ready.
2. The card has an Internet (`inet`) address (in this case `192.168.1.3`).
3. It has a valid subnet mask (`netmask`; `0xffffffff` is the same as `255.255.255.0`).
4. It has a valid broadcast address (in this case, `192.168.1.255`).
5. The MAC address of the card (`ether`) is `00:a0:cc:da:da:da`
6. The physical media selection is on autoselection mode (`media: Ethernet autoselect (100baseTX <full-duplex>)`). We see that `dc1` was configured to run with `10baseT/UTP` media. For more information on available media types for a driver, please refer to its manual page.
7. The status of the link (`status`) is `active`, i.e. the carrier is detected. For `dc1`, we see `status: no carrier`. This is normal when an Ethernet cable is not plugged into the card.

If the `ifconfig(8)` output had shown something similar to:

```

dc0: flags=8843<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
    ether 00:a0:cc:da:da:da

```

it would indicate the card has not been configured.

To configure your card, you need `root` privileges. The network card configuration can be done from the command line with `ifconfig(8)` but you would have to do it after each reboot of the system. The file `/etc/rc.conf` is where to add the network card's configuration.

Open `/etc/rc.conf` in your favorite editor. You need to add a line for each network card present on the system, for example in our case, we added these lines:

```
ifconfig_dc0="inet 192.168.1.3 netmask 255.255.255.0"
ifconfig_dc1="inet 10.0.0.1 netmask 255.255.255.0 media 10baseT/UTP"
```

You have to replace `dc0`, `dc1`, and so on, with the correct device for your cards, and the addresses with the proper ones. You should read the card driver and `ifconfig(8)` manual pages for more details about the allowed options and also `rc.conf(5)` manual page for more information on the syntax of `/etc/rc.conf`.

If you configured the network during installation, some lines about the network card(s) may be already present. Double check `/etc/rc.conf` before adding any lines.

You will also have to edit the file `/etc/hosts` to add the names and the IP addresses of various machines of the LAN, if they are not already there. For more information please refer to `hosts(5)` and to `/usr/share/examples/etc/hosts`.

11.8.3. Testing and Troubleshooting

Once you have made the necessary changes in `/etc/rc.conf`, you should reboot your system. This will allow the change(s) to the interface(s) to be applied, and verify that the system restarts without any configuration errors.

Once the system has been rebooted, you should test the network interfaces.

11.8.3.1. Testing the Ethernet Card

To verify that an Ethernet card is configured correctly, you have to try two things. First, ping the interface itself, and then ping another machine on the LAN.

First test the local interface:

```
% ping -c5 192.168.1.3
PING 192.168.1.3 (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: icmp_seq=0 ttl=64 time=0.082 ms
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.108 ms
64 bytes from 192.168.1.3: icmp_seq=4 ttl=64 time=0.076 ms

--- 192.168.1.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.074/0.083/0.108/0.013 ms
```

Now we have to ping another machine on the LAN:

```
% ping -c5 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=0.726 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.766 ms
```

```

64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.700 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.747 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.704 ms

--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.700/0.729/0.766/0.025 ms

```

You could also use the machine name instead of `192.168.1.2` if you have set up the `/etc/hosts` file.

11.8.3.2. Troubleshooting

Troubleshooting hardware and software configurations is always a pain, and a pain which can be alleviated by checking the simple things first. Is your network cable plugged in? Have you properly configured the network services? Did you configure the firewall correctly? Is the card you are using supported by FreeBSD? Always check the hardware notes before sending off a bug report. Update your version of FreeBSD to the latest STABLE version. Check the mailing list archives, or perhaps search the Internet.

If the card works, yet performance is poor, it would be worthwhile to read over the `tuning(7)` manual page. You can also check the network configuration as incorrect network settings can cause slow connections.

Some users experience one or two `device timeout` messages, which is normal for some cards. If they continue, or are bothersome, you may wish to be sure the device is not conflicting with another device. Double check the cable connections. Perhaps you may just need to get another card.

At times, users see a few `watchdog timeout` errors. The first thing to do here is to check your network cable. Many cards require a PCI slot which supports Bus Mastering. On some old motherboards, only one PCI slot allows it (usually slot 0). Check the network card and the motherboard documentation to determine if that may be the problem.

`No route to host` messages occur if the system is unable to route a packet to the destination host. This can happen if no default route is specified, or if a cable is unplugged. Check the output of `netstat -rn` and make sure there is a valid route to the host you are trying to reach. If there is not, read on to Rozdział 27.

`ping: sendto: Permission denied` error messages are often caused by a misconfigured firewall. If `ipfw` is enabled in the kernel but no rules have been defined, then the default policy is to deny all traffic, even ping requests! Read on to Rozdział 26 for more information.

Sometimes performance of the card is poor, or below average. In these cases it is best to set the media selection mode from `autoselect` to the correct media selection. While this usually works for most hardware, it may not resolve this issue for everyone. Again, check all the network settings, and read over the `tuning(7)` manual page.

11.9. Virtual Hosts

A very common use of FreeBSD is virtual site hosting, where one server appears to the network as many servers. This is achieved by assigning multiple network addresses to a single interface.

A given network interface has one „real” address, and may have any number of „alias” addresses. These aliases are normally added by placing alias entries in `/etc/rc.conf`.

An alias entry for the interface `fxp0` looks like:

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

Note that alias entries must start with `alias0` and proceed upwards in order, (for example, `_alias1`, `_alias2`, and so on). The configuration process will stop at the first missing number.

The calculation of alias netmasks is important, but fortunately quite simple. For a given interface, there must be one address which correctly represents the network's netmask. Any other addresses which fall within this network must have a netmask of all 1s (expressed as either `255.255.255.255` or `0xffffffff`).

For example, consider the case where the `fxp0` interface is connected to two networks, the `10.1.1.0` network with a netmask of `255.255.255.0` and the `202.0.75.16` network with a netmask of `255.255.255.240`. We want the system to appear at `10.1.1.1` through `10.1.1.5` and at `202.0.75.17` through `202.0.75.20`. As noted above, only the first address in a given network range (in this case, `10.1.1.1` and `202.0.75.17`) should have a real netmask; all the rest (`10.1.1.2` through `10.1.1.5` and `202.0.75.18` through `202.0.75.20`) must be configured with a netmask of `255.255.255.255`.

The following `/etc/rc.conf` entries configure the adapter correctly for this arrangement:

```
ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
ifconfig_fxp0_alias0="inet 10.1.1.2 netmask 255.255.255.255"
ifconfig_fxp0_alias1="inet 10.1.1.3 netmask 255.255.255.255"
ifconfig_fxp0_alias2="inet 10.1.1.4 netmask 255.255.255.255"
ifconfig_fxp0_alias3="inet 10.1.1.5 netmask 255.255.255.255"
ifconfig_fxp0_alias4="inet 202.0.75.17 netmask 255.255.255.240"
ifconfig_fxp0_alias5="inet 202.0.75.18 netmask 255.255.255.255"
ifconfig_fxp0_alias6="inet 202.0.75.19 netmask 255.255.255.255"
ifconfig_fxp0_alias7="inet 202.0.75.20 netmask 255.255.255.255"
```

11.10. Configuration Files

11.10.1. /etc Layout

There are a number of directories in which configuration information is kept. These include:

<code>/etc</code>	Generic system configuration information; data here is system-specific.
<code>/etc/defaults</code>	Default versions of system configuration files.
<code>/etc/mail</code>	Extra sendmail(8) configuration, other MTA configuration files.
<code>/etc/ppp</code>	Configuration for both user- and kernel-ppp programs.
<code>/etc/namedb</code>	Default location for named(8) data. Normally <code>named.conf</code> and zone files are stored here.
<code>/usr/local/etc</code>	Configuration files for installed applications. May contain per-application subdirectories.
<code>/usr/local/etc/rc.d</code>	Start/stop scripts for installed applications.
<code>/var/db</code>	Automatically generated system-specific database files, such as the package database, the locate database, and so on

11.10.2. Hostnames

11.10.2.1. `/etc/resolv.conf`

`/etc/resolv.conf` dictates how FreeBSD's resolver accesses the Internet Domain Name System (DNS).

The most common entries to `resolv.conf` are:

<code>nameserver</code>	The IP address of a name server the resolver should query. The servers are queried in the order listed with a maximum of three.
<code>search</code>	Search list for hostname lookup. This is normally determined by the domain of the local hostname.
<code>domain</code>	The local domain name.

A typical `resolv.conf`:

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

Notatka: Only one of the `search` and `domain` options should be used.

If you are using DHCP, `dhclient(8)` usually rewrites `resolv.conf` with information received from the DHCP server.

11.10.2.2. `/etc/hosts`

`/etc/hosts` is a simple text database reminiscent of the old Internet. It works in conjunction with DNS and NIS providing name to IP address mappings. Local computers connected via a LAN can be placed in here for simplistic naming purposes instead of setting up a `named(8)` server. Additionally, `/etc/hosts` can be used to provide a local record of Internet names, reducing the need to query externally for commonly accessed names.

```
# $FreeBSD$
#
# Host Database
# This file should contain the addresses and aliases
# for local hosts that share this file.
# In the presence of the domain name service or NIS, this file may
# not be consulted at all; see /etc/nsswitch.conf for the resolution order.
#
#
::1                localhost localhost.my.domain myname.my.domain
127.0.0.1          localhost localhost.my.domain myname.my.domain
#
# Imaginary network.
#10.0.0.2          myname.my.domain myname
#10.0.0.3          myfriend.my.domain myfriend
#
# According to RFC 1918, you can use the following IP networks for
```

```
# private nets which will never be connected to the Internet:
#
#      10.0.0.0          -   10.255.255.255
#      172.16.0.0       -   172.31.255.255
#      192.168.0.0      -   192.168.255.255
#
# In case you want to be able to connect to the Internet, you need
# real official assigned numbers. PLEASE PLEASE PLEASE do not try
# to invent your own network numbers but instead get one from your
# network provider (if any) or from the Internet Registry (ftp to
# rs.internic.net, directory '/templates').
#
```

/etc/hosts takes on the simple format of:

```
[Internet address] [official hostname] [alias1] [alias2] ...
```

For example:

```
10.0.0.1 myRealHostname.example.com myRealHostname foobar1 foobar2
```

Consult hosts(5) for more information.

11.10.3. Log File Configuration

11.10.3.1. syslog.conf

syslog.conf is the configuration file for the syslogd(8) program. It indicates which types of syslog messages are logged to particular log files.

```
# $FreeBSD$
#
#      Spaces ARE valid field separators in this file. However,
#      other *nix-like systems still insist on using tabs as field
#      separators. If you are sharing this file between systems, you
#      may want to use only tabs as field separators here.
#      Consult the syslog.conf(5) manual page.
*.err;kern.debug;auth.notice;mail.crit      /dev/console
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages
security.*                                   /var/log/security
mail.info                                    /var/log/maillog
lpr.info                                     /var/log/lpd-errs
cron.*                                       /var/log/cron
*.err                                         root
*.notice;news.err                           root
*.alert                                       root
*.emerg                                       *
# uncomment this to log all writes to /dev/console to /var/log/console.log
#console.info                                /var/log/console.log
# uncomment this to enable logging of all log messages to /var/log/all.log
#*. *                                         /var/log/all.log
```

```
# uncomment this to enable logging to a remote log host named loghost
#*. *                               @loghost
# uncomment these if you're running inn
# news.crit                         /var/log/news/news.crit
# news.err                          /var/log/news/news.err
# news.notice                       /var/log/news/news.notice
!startslip
*. *                               /var/log/slip.log
!ppp
*. *                               /var/log/ppp.log
```

Consult the syslog.conf(5) manual page for more information.

11.10.3.2. newsyslog.conf

newsyslog.conf is the configuration file for newsyslog(8), a program that is normally scheduled to run by cron(8). newsyslog(8) determines when log files require archiving or rearranging. logfile is moved to logfile.0, logfile.0 is moved to logfile.1, and so on. Alternatively, the log files may be archived in gzip(1) format causing them to be named: logfile.0.gz, logfile.1.gz, and so on.

newsyslog.conf indicates which log files are to be managed, how many are to be kept, and when they are to be touched. Log files can be rearranged and/or archived when they have either reached a certain size, or at a certain periodic time/date.

```
# configuration file for newsyslog
# $FreeBSD$
#
# filename           [owner:group]    mode count size when [ZB] [/pid_file] [sig_num]
/var/log/cron        600 3      100 *      Z
/var/log/amd.log      644 7      100 *      Z
/var/log/kerberos.log 644 7      100 *      Z
/var/log/lpd-errs     644 7      100 *      Z
/var/log/maillog      644 7      *    @T00    Z
/var/log/sendmail.st  644 10     *    168    B
/var/log/messages     644 5      100 *      Z
/var/log/all.log      600 7      *    @T00    Z
/var/log/slip.log     600 3      100 *      Z
/var/log/ppp.log      600 3      100 *      Z
/var/log/security     600 10     100 *      Z
/var/log/wtmp         644 3      *    @01T05 B
/var/log/daily.log    640 7      *    @T00    Z
/var/log/weekly.log   640 5      1    $W6D0   Z
/var/log/monthly.log  640 12     *    $M1D0   Z
/var/log/console.log  640 5      100 *      Z
```

Consult the newsyslog(8) manual page for more information.

11.10.4. sysctl.conf

`sysctl.conf` looks much like `rc.conf`. Values are set in a `variable=value` form. The specified values are set after the system goes into multi-user mode. Not all variables are settable in this mode.

A sample `sysctl.conf` turning off logging of fatal signal exits and letting Linux programs know they are really running under FreeBSD:

```
kern.logsigexit=0      # Do not log fatal signal exits (e.g. sig 11)
compat.linux.osname=FreeBSD
compat.linux.osrelease=4.3-STABLE
```

11.11. Tuning with sysctl

`sysctl(8)` is an interface that allows you to make changes to a running FreeBSD system. This includes many advanced options of the TCP/IP stack and virtual memory system that can dramatically improve performance for an experienced system administrator. Over five hundred system variables can be read and set using `sysctl(8)`.

At its core, `sysctl(8)` serves two functions: to read and to modify system settings.

To view all readable variables:

```
% sysctl -a
```

To read a particular variable, for example, `kern.maxproc`:

```
% sysctl kern.maxproc
kern.maxproc: 1044
```

To set a particular variable, use the intuitive `variable=value` syntax:

```
# sysctl kern.maxfiles=5000
kern.maxfiles: 2088 -> 5000
```

Settings of `sysctl` variables are usually either strings, numbers, or booleans (a boolean being 1 for yes or a 0 for no).

If you want to set automatically some variables each time the machine boots, add them to the `/etc/sysctl.conf` file. For more information see the `sysctl.conf(5)` manual page and the Sekcja 11.10.4.

11.11.1. sysctl(8) Read-only

Contributed by Tom Rhodes.

In some cases it may be desirable to modify read-only `sysctl(8)` values. While this is sometimes unavoidable, it can only be done on (re)boot.

For instance on some laptop models the `cardbus(4)` device will not probe memory ranges, and fail with errors which look similar to:

```
cbb0: Could not map register memory
device_probe_and_attach: cbb0 attach returned 12
```

Cases like the one above usually require the modification of some default `sysctl(8)` settings which are set read only. To overcome these situations a user can put `sysctl(8)` „OIDs” in their local `/boot/loader.conf`. Default settings are located in the `/boot/defaults/loader.conf` file.

Fixing the problem mentioned above would require a user to set `hw.pci.allow_unsupported_io_range=1` in the aforementioned file. Now `cardbus(4)` will work properly.

11.12. Tuning Disks

11.12.1. Sysctl Variables

11.12.1.1. `vfs.vmiodirenable`

The `vfs.vmiodirenable` `sysctl` variable may be set to either 0 (off) or 1 (on); it is 1 by default. This variable controls how directories are cached by the system. Most directories are small, using just a single fragment (typically 1 K) in the file system and less (typically 512 bytes) in the buffer cache. With this variable turned off (to 0), the buffer cache will only cache a fixed number of directories even if you have a huge amount of memory. When turned on (to 1), this `sysctl` allows the buffer cache to use the VM Page Cache to cache the directories, making all the memory available for caching directories. However, the minimum in-core memory used to cache a directory is the physical page size (typically 4 K) rather than 512 bytes. We recommend keeping this option on if you are running any services which manipulate large numbers of files. Such services can include web caches, large mail systems, and news systems. Keeping this option on will generally not reduce performance even with the wasted memory but you should experiment to find out.

11.12.1.2. `vfs.write_behind`

The `vfs.write_behind` `sysctl` variable defaults to 1 (on). This tells the file system to issue media writes as full clusters are collected, which typically occurs when writing large sequential files. The idea is to avoid saturating the buffer cache with dirty buffers when it would not benefit I/O performance. However, this may stall processes and under certain circumstances you may wish to turn it off.

11.12.1.3. `vfs.hirunningspace`

The `vfs.hirunningspace` `sysctl` variable determines how much outstanding write I/O may be queued to disk controllers system-wide at any given instance. The default is usually sufficient but on machines with lots of disks you may want to bump it up to four or five *megabytes*. Note that setting too high a value (exceeding the buffer cache’s write threshold) can lead to extremely bad clustering performance. Do not set this value arbitrarily high! Higher write values may add latency to reads occurring at the same time.

There are various other buffer-cache and VM page cache related `sysctls`. We do not recommend modifying these values, the VM system does an extremely good job of automatically tuning itself.

11.12.1.4. vm.swap_idle_enabled

The `vm.swap_idle_enabled` sysctl variable is useful in large multi-user systems where you have lots of users entering and leaving the system and lots of idle processes. Such systems tend to generate a great deal of continuous pressure on free memory reserves. Turning this feature on and tweaking the swapout hysteresis (in idle seconds) via `vm.swap_idle_threshold1` and `vm.swap_idle_threshold2` allows you to depress the priority of memory pages associated with idle processes more quickly than the normal pageout algorithm. This gives a helping hand to the pageout daemon. Do not turn this option on unless you need it, because the tradeoff you are making is essentially pre-page memory sooner rather than later; thus eating more swap and disk bandwidth. In a small system this option will have a determinable effect but in a large system that is already doing moderate paging this option allows the VM system to stage whole processes into and out of memory easily.

11.12.1.5. hw.ata.wc

FreeBSD 4.3 flirted with turning off IDE write caching. This reduced write bandwidth to IDE disks but was considered necessary due to serious data consistency issues introduced by hard drive vendors. The problem is that IDE drives lie about when a write completes. With IDE write caching turned on, IDE hard drives not only write data to disk out of order, but will sometimes delay writing some blocks indefinitely when under heavy disk loads. A crash or power failure may cause serious file system corruption. FreeBSD's default was changed to be safe. Unfortunately, the result was such a huge performance loss that we changed write caching back to on by default after the release. You should check the default on your system by observing the `hw.ata.wc` sysctl variable. If IDE write caching is turned off, you can turn it back on by setting the kernel variable back to 1. This must be done from the boot loader at boot time. Attempting to do it after the kernel boots will have no effect.

For more information, please see `ata(4)`.

11.12.1.6. SCSI_DELAY (kern.cam.scsi_delay)

The `SCSI_DELAY` kernel config may be used to reduce system boot times. The defaults are fairly high and can be responsible for 15 seconds of delay in the boot process. Reducing it to 5 seconds usually works (especially with modern drives). Newer versions of FreeBSD (5.0 and higher) should use the `kern.cam.scsi_delay` boot time tunable. The tunable, and kernel config option accept values in terms of *milliseconds* and *not seconds*.

11.12.2. Soft Updates

The `tunefs(8)` program can be used to fine-tune a file system. This program has many different options, but for now we are only concerned with toggling Soft Updates on and off, which is done by:

```
# tunefs -n enable /filesystem
# tunefs -n disable /filesystem
```

A filesystem cannot be modified with `tunefs(8)` while it is mounted. A good time to enable Soft Updates is before any partitions have been mounted, in single-user mode.

Soft Updates drastically improves meta-data performance, mainly file creation and deletion, through the use of a memory cache. We recommend to use Soft Updates on all of your file systems. There are two downsides to Soft Updates that you should be aware of: First, Soft Updates guarantees filesystem consistency in the case of a crash but could very easily be several seconds (even a minute!) behind updating the physical disk. If your system crashes you

may lose more work than otherwise. Secondly, Soft Updates delays the freeing of filesystem blocks. If you have a filesystem (such as the root filesystem) which is almost full, performing a major update, such as `make installworld`, can cause the filesystem to run out of space and the update to fail.

11.12.2.1. More Details about Soft Updates

There are two traditional approaches to writing a file systems meta-data back to disk. (Meta-data updates are updates to non-content data like inodes or directories.)

Historically, the default behavior was to write out meta-data updates synchronously. If a directory had been changed, the system waited until the change was actually written to disk. The file data buffers (file contents) were passed through the buffer cache and backed up to disk later on asynchronously. The advantage of this implementation is that it operates safely. If there is a failure during an update, the meta-data are always in a consistent state. A file is either created completely or not at all. If the data blocks of a file did not find their way out of the buffer cache onto the disk by the time of the crash, `fsck(8)` is able to recognize this and repair the filesystem by setting the file length to 0. Additionally, the implementation is clear and simple. The disadvantage is that meta-data changes are slow. An `rm -r`, for instance, touches all the files in a directory sequentially, but each directory change (deletion of a file) will be written synchronously to the disk. This includes updates to the directory itself, to the inode table, and possibly to indirect blocks allocated by the file. Similar considerations apply for unrolling large hierarchies (`tar -x`).

The second case is asynchronous meta-data updates. This is the default for Linux/ext2fs and `mount -o async` for *BSD ufs. All meta-data updates are simply being passed through the buffer cache too, that is, they will be intermixed with the updates of the file content data. The advantage of this implementation is there is no need to wait until each meta-data update has been written to disk, so all operations which cause huge amounts of meta-data updates work much faster than in the synchronous case. Also, the implementation is still clear and simple, so there is a low risk for bugs creeping into the code. The disadvantage is that there is no guarantee at all for a consistent state of the filesystem. If there is a failure during an operation that updated large amounts of meta-data (like a power failure, or someone pressing the reset button), the filesystem will be left in an unpredictable state. There is no opportunity to examine the state of the filesystem when the system comes up again; the data blocks of a file could already have been written to the disk while the updates of the inode table or the associated directory were not. It is actually impossible to implement a `fsck` which is able to clean up the resulting chaos (because the necessary information is not available on the disk). If the filesystem has been damaged beyond repair, the only choice is to use `newfs(8)` on it and restore it from backup.

The usual solution for this problem was to implement *dirty region logging*, which is also referred to as *journaling*, although that term is not used consistently and is occasionally applied to other forms of transaction logging as well. Meta-data updates are still written synchronously, but only into a small region of the disk. Later on they will be moved to their proper location. Because the logging area is a small, contiguous region on the disk, there are no long distances for the disk heads to move, even during heavy operations, so these operations are quicker than synchronous updates. Additionally the complexity of the implementation is fairly limited, so the risk of bugs being present is low. A disadvantage is that all meta-data are written twice (once into the logging region and once to the proper location) so for normal work, a performance „pessimization” might result. On the other hand, in case of a crash, all pending meta-data operations can be quickly either rolled-back or completed from the logging area after the system comes up again, resulting in a fast filesystem startup.

Kirk McKusick, the developer of Berkeley FFS, solved this problem with Soft Updates: all pending meta-data updates are kept in memory and written out to disk in a sorted sequence („ordered meta-data updates”). This has the effect that, in case of heavy meta-data operations, later updates to an item „catch” the earlier ones if the earlier ones are still in memory and have not already been written to disk. So all operations on, say, a directory are generally performed in memory before the update is written to disk (the data blocks are sorted according to their position so

that they will not be on the disk ahead of their meta-data). If the system crashes, this causes an implicit „log rewind”: all operations which did not find their way to the disk appear as if they had never happened. A consistent filesystem state is maintained that appears to be the one of 30 to 60 seconds earlier. The algorithm used guarantees that all resources in use are marked as such in their appropriate bitmaps: blocks and inodes. After a crash, the only resource allocation error that occurs is that resources are marked as „used” which are actually „free”. `fsck(8)` recognizes this situation, and frees the resources that are no longer used. It is safe to ignore the dirty state of the filesystem after a crash by forcibly mounting it with `mount -f`. In order to free resources that may be unused, `fsck(8)` needs to be run at a later time. This is the idea behind the *background fsck*: at system startup time, only a *snapshot* of the filesystem is recorded. The `fsck` can be run later on. All file systems can then be mounted „dirty”, so the system startup proceeds in multiuser mode. Then, *background fscks* will be scheduled for all file systems where this is required, to free resources that may be unused. (File systems that do not use Soft Updates still need the usual foreground `fsck` though.)

The advantage is that meta-data operations are nearly as fast as asynchronous updates (i.e. faster than with *logging*, which has to write the meta-data twice). The disadvantages are the complexity of the code (implying a higher risk for bugs in an area that is highly sensitive regarding loss of user data), and a higher memory consumption. Additionally there are some idiosyncrasies one has to get used to. After a crash, the state of the filesystem appears to be somewhat „older”. In situations where the standard synchronous approach would have caused some zero-length files to remain after the `fsck`, these files do not exist at all with a Soft Updates filesystem because neither the meta-data nor the file contents have ever been written to disk. Disk space is not released until the updates have been written to disk, which may take place some time after running `rm`. This may cause problems when installing large amounts of data on a filesystem that does not have enough free space to hold all the files twice.

11.13. Tuning Kernel Limits

11.13.1. File/Process Limits

11.13.1.1. `kern.maxfiles`

`kern.maxfiles` can be raised or lowered based upon your system requirements. This variable indicates the maximum number of file descriptors on your system. When the file descriptor table is full, `file: table is full` will show up repeatedly in the system message buffer, which can be viewed with the `dmesg` command.

Each open file, socket, or fifo uses one file descriptor. A large-scale production server may easily require many thousands of file descriptors, depending on the kind and number of services running concurrently.

In older FreeBSD releases, `kern.maxfile`’s default value is derived from the `maxusers` option in your kernel configuration file. `kern.maxfiles` grows proportionally to the value of `maxusers`. When compiling a custom kernel, it is a good idea to set this kernel configuration option according to the uses of your system. From this number, the kernel is given most of its pre-defined limits. Even though a production machine may not actually have 256 users connected at once, the resources needed may be similar to a high-scale web server.

As of FreeBSD 4.5, `kern.maxusers` is automatically sized at boot based on the amount of memory available in the system, and may be determined at run-time by inspecting the value of the read-only `kern.maxusers` sysctl. Some sites will require larger or smaller values of `kern.maxusers` and may set it as a loader tunable; values of 64, 128, and 256 are not uncommon. We do not recommend going above 256 unless you need a huge number of file descriptors; many of the tunable values set to their defaults by `kern.maxusers` may be individually overridden at

boot-time or run-time in `/boot/loader.conf` (see the `loader.conf(5)` man page or the `/boot/defaults/loader.conf` file for some hints) or as described elsewhere in this document. Systems older than FreeBSD 4.4 must set this value via the kernel `config(8)` option `maxusers` instead.

In older releases, the system will auto-tune `maxusers` for you if you explicitly set it to 0¹. When setting this option, you will want to set `maxusers` to at least 4, especially if you are using the X Window System or compiling software. The reason is that the most important table set by `maxusers` is the maximum number of processes, which is set to $20 + 16 * \text{maxusers}$, so if you set `maxusers` to 1, then you can only have 36 simultaneous processes, including the 18 or so that the system starts up at boot time and the 15 or so you will probably create when you start the X Window System. Even a simple task like reading a manual page will start up nine processes to filter, decompress, and view it. Setting `maxusers` to 64 will allow you to have up to 1044 simultaneous processes, which should be enough for nearly all uses. If, however, you see the dreaded `proc table full` error when trying to start another program, or are running a server with a large number of simultaneous users (like `ftp.FreeBSD.org`), you can always increase the number and rebuild.

Notatka: `maxusers` does *not* limit the number of users which can log into your machine. It simply sets various table sizes to reasonable values considering the maximum number of users you will likely have on your system and how many processes each of them will be running. One keyword which *does* limit the number of simultaneous remote logins and X terminal windows is `pseudo-device pty 16`. With FreeBSD 5.X, you do not have to worry about this number since the `pty(4)` driver is „auto-cloning”; you simply use the line `device pty` in your configuration file.

11.13.1.2. kern.ipc.somaxconn

The `kern.ipc.somaxconn` `sysctl` variable limits the size of the listen queue for accepting new TCP connections. The default value of 128 is typically too low for robust handling of new connections in a heavily loaded web server environment. For such environments, it is recommended to increase this value to 1024 or higher. The service daemon may itself limit the listen queue size (e.g. `sendmail(8)`, or **Apache**) but will often have a directive in its configuration file to adjust the queue size. Large listen queues also do a better job of avoiding Denial of Service (DoS) attacks.

11.13.2. Network Limits

The `NMBCLUSTERS` kernel configuration option dictates the amount of network Mbufs available to the system. A heavily-trafficked server with a low number of Mbufs will hinder FreeBSD’s ability. Each cluster represents approximately 2 K of memory, so a value of 1024 represents 2 megabytes of kernel memory reserved for network buffers. A simple calculation can be done to figure out how many are needed. If you have a web server which maxes out at 1000 simultaneous connections, and each connection eats a 16 K receive and 16 K send buffer, you need approximately 32 MB worth of network buffers to cover the web server. A good rule of thumb is to multiply by 2, so $2 \times 32 \text{ MB} / 2 \text{ KB} = 64 \text{ MB} / 2 \text{ kB} = 32768$. We recommend values between 4096 and 32768 for machines with greater amounts of memory. Under no circumstances should you specify an arbitrarily high value for this parameter as it could lead to a boot time crash. The `-m` option to `netstat(1)` may be used to observe network cluster use.

`kern.ipc.nmbclusters` loader tunable should be used to tune this at boot time. Only older versions of FreeBSD will require you to use the `NMBCLUSTERS` kernel `config(8)` option.

For busy servers that make extensive use of the `sendfile(2)` system call, it may be necessary to increase the number of `sendfile(2)` buffers via the `NSFBUFS` kernel configuration option or by setting its value in `/boot/loader.conf` (see

loader(8) for details). A common indicator that this parameter needs to be adjusted is when processes are seen in the `sfbufa` state. The `sysctl` variable `kern.ipc.nsfbufs` is a read-only glimpse at the kernel configured variable. This parameter nominally scales with `kern.maxusers`, however it may be necessary to tune accordingly.

WAŻNE: Even though a socket has been marked as non-blocking, calling `sendfile(2)` on the non-blocking socket may result in the `sendfile(2)` call blocking until enough `struct sf_buf`'s are made available.

11.13.2.1. `net.inet.ip.portrange.*`

The `net.inet.ip.portrange.*` `sysctl` variables control the port number ranges automatically bound to TCP and UDP sockets. There are three ranges: a low range, a default range, and a high range. Most network programs use the default range which is controlled by the `net.inet.ip.portrange.first` and `net.inet.ip.portrange.last`, which default to 1024 and 5000, respectively. Bound port ranges are used for outgoing connections, and it is possible to run the system out of ports under certain circumstances. This most commonly occurs when you are running a heavily loaded web proxy. The port range is not an issue when running servers which handle mainly incoming connections, such as a normal web server, or has a limited number of outgoing connections, such as a mail relay. For situations where you may run yourself out of ports, it is recommended to increase `net.inet.ip.portrange.last` modestly. A value of 10000, 20000 or 30000 may be reasonable. You should also consider firewall effects when changing the port range. Some firewalls may block large ranges of ports (usually low-numbered ports) and expect systems to use higher ranges of ports for outgoing connections — for this reason it is not recommended that `net.inet.ip.portrange.first` be lowered.

11.13.2.2. TCP Bandwidth Delay Product

The TCP Bandwidth Delay Product Limiting is similar to TCP/Vegas in NetBSD. It can be enabled by setting `net.inet.tcp.inflight.enable` `sysctl` variable to 1. The system will attempt to calculate the bandwidth delay product for each connection and limit the amount of data queued to the network to just the amount required to maintain optimum throughput.

This feature is useful if you are serving data over modems, Gigabit Ethernet, or even high speed WAN links (or any other link with a high bandwidth delay product), especially if you are also using window scaling or have configured a large send window. If you enable this option, you should also be sure to set `net.inet.tcp.inflight.debug` to 0 (disable debugging), and for production use setting `net.inet.tcp.inflight.min` to at least 6144 may be beneficial. However, note that setting high minimums may effectively disable bandwidth limiting depending on the link. The limiting feature reduces the amount of data built up in intermediate route and switch packet queues as well as reduces the amount of data built up in the local host's interface queue. With fewer packets queued up, interactive connections, especially over slow modems, will also be able to operate with lower *Round Trip Times*. However, note that this feature only effects data transmission (uploading / server side). It has no effect on data reception (downloading).

Adjusting `net.inet.tcp.inflight.stab` is *not* recommended. This parameter defaults to 20, representing 2 maximal packets added to the bandwidth delay product window calculation. The additional window is required to stabilize the algorithm and improve responsiveness to changing conditions, but it can also result in higher ping times over slow links (though still much lower than you would get without the `inflight` algorithm). In such cases, you may wish to try reducing this parameter to 15, 10, or 5; and may also have to reduce `net.inet.tcp.inflight.min` (for example, to 3500) to get the desired effect. Reducing these parameters should be done as a last resort only.

11.13.3. Virtual Memory

11.13.3.1. `kern.maxvnodes`

A vnode is the internal representation of a file or directory. So increasing the number of vnodes available to the operating system cuts down on disk I/O. Normally this is handled by the operating system and does not need to be changed. In some cases where disk I/O is a bottleneck and the system is running out of vnodes, this setting will need to be increased. The amount of inactive and free RAM will need to be taken into account.

To see the current number of vnodes in use:

```
# sysctl vfs.numvnodes
vfs.numvnodes: 91349
```

To see the maximum vnodes:

```
# sysctl kern.maxvnodes
kern.maxvnodes: 100000
```

If the current vnode usage is near the maximum, increasing `kern.maxvnodes` by a value of 1,000 is probably a good idea. Keep an eye on the number of `vfs.numvnodes`. If it climbs up to the maximum again, `kern.maxvnodes` will need to be increased further. A shift in your memory usage as reported by `top(1)` should be visible. More memory should be active.

11.14. Adding Swap Space

No matter how well you plan, sometimes a system does not run as you expect. If you find you need more swap space, it is simple enough to add. You have three ways to increase swap space: adding a new hard drive, enabling swap over NFS, and creating a swap file on an existing partition.

For information on how to encrypt swap space, what options for this task exist and why it should be done, please refer to Sekcja 17.17 of the Handbook.

11.14.1. Swap on a New Hard Drive

The best way to add swap, of course, is to use this as an excuse to add another hard drive. You can always use another hard drive, after all. If you can do this, go reread the discussion of swap space in Sekcja 11.2 of the Handbook for some suggestions on how to best arrange your swap.

11.14.2. Swapping over NFS

Swapping over NFS is only recommended if you do not have a local hard disk to swap to; NFS swapping will be limited by the available network bandwidth and puts an additional burden on the NFS server.

11.14.3. Swapfiles

You can create a file of a specified size to use as a swap file. In our example here we will use a 64MB file called `/usr/swap0`. You can use any name you want, of course.

Przykład 11-1. Creating a Swapfile on FreeBSD

1. Be certain that your kernel configuration includes the memory disk driver (`md(4)`). It is default in `GENERIC` kernel.

```
device    md      # Memory "disks"
```
2. Create a swapfile (`/usr/swap0`):

```
# dd if=/dev/zero of=/usr/swap0 bs=1024k count=64
```
3. Set proper permissions on (`/usr/swap0`):

```
# chmod 0600 /usr/swap0
```
4. Enable the swap file in `/etc/rc.conf`:

```
swapfile="/usr/swap0"    # Set to name of swapfile if aux swapfile desired.
```
5. Reboot the machine or to enable the swap file immediately, type:

```
# mdconfig -a -t vnode -f /usr/swap0 -u 0 && swapon /dev/md0
```

11.15. Power and Resource Management

Written by Hiten Pandya i Tom Rhodes.

It is very important to utilize hardware resources in an efficient manner. Before ACPI was introduced, it was very difficult and inflexible for operating systems to manage the power usage and thermal properties of a system. The hardware was controlled by some sort of BIOS embedded interface, such as *Plug and Play BIOS (PNPBIOS)*, or *Advanced Power Management (APM)* and so on. Power and Resource Management is one of the key components of a modern operating system. For example, you may want an operating system to monitor system limits (and possibly alert you) in case your system temperature increased unexpectedly.

In this section of the FreeBSD Handbook, we will provide comprehensive information about ACPI. References will be provided for further reading at the end.

11.15.1. What Is ACPI?

Advanced Configuration and Power Interface (ACPI) is a standard written by an alliance of vendors to provide a standard interface for hardware resources and power management (hence the name). It is a key element in *Operating System-directed configuration and Power Management*, i.e.: it provides more control and flexibility to the operating system (OS). Modern systems „stretched” the limits of the current Plug and Play interfaces prior to the introduction of ACPI. ACPI is the direct successor to APM (Advanced Power Management).

11.15.2. Shortcomings of Advanced Power Management (APM)

The *Advanced Power Management (APM)* facility controls the power usage of a system based on its activity. The APM BIOS is supplied by the (system) vendor and it is specific to the hardware platform. An APM driver in the OS mediates access to the *APM Software Interface*, which allows management of power levels.

There are four major problems in APM. Firstly, power management is done by the (vendor-specific) BIOS, and the OS does not have any knowledge of it. One example of this, is when the user sets idle-time values for a hard drive in the APM BIOS, that when exceeded, it (BIOS) would spin down the hard drive, without the consent of the OS. Secondly, the APM logic is embedded in the BIOS, and it operates outside the scope of the OS. This means users can only fix problems in their APM BIOS by flashing a new one into the ROM; which is a very dangerous procedure with the potential to leave the system in an unrecoverable state if it fails. Thirdly, APM is a vendor-specific technology, which means that there is a lot of parity (duplication of efforts) and bugs found in one vendor's BIOS, may not be solved in others. Last but not the least, the APM BIOS did not have enough room to implement a sophisticated power policy, or one that can adapt very well to the purpose of the machine.

Plug and Play BIOS (PNPBIOS) was unreliable in many situations. PNPBIOS is 16-bit technology, so the OS has to use 16-bit emulation in order to „interface” with PNPBIOS methods.

The FreeBSD APM driver is documented in the apm(4) manual page.

11.15.3. Configuring ACPI

The `acpi.ko` driver is loaded by default at start up by the loader(8) and should *not* be compiled into the kernel. The reasoning behind this is that modules are easier to work with, say if switching to another `acpi.ko` without doing a kernel rebuild. This has the advantage of making testing easier. Another reason is that starting ACPI after a system has been brought up is not too useful, and in some cases can be fatal. In doubt, just disable ACPI all together. This driver should not and can not be unloaded because the system bus uses it for various hardware interactions. ACPI can be disabled with the `acpicnf(8)` utility. In fact most of the interaction with ACPI can be done via `acpicnf(8)`. Basically this means, if anything about ACPI is in the `dmesg(8)` output, then most likely it is already running.

Notatka: ACPI and APM cannot coexist and should be used separately. The last one to load will terminate if the driver notices the other running.

In the simplest form, ACPI can be used to put the system into a sleep mode with `acpicnf(8)`, the `-s` flag, and a 1-5 option. Most users will only need 1. Option 5 will do a soft-off which is the same action as:

```
# halt -p
```

The other options are available. Check out the `acpicnf(8)` manual page for more information.

11.16. Using and Debugging FreeBSD ACPI

Written by Nate Lawson. With contributions from Peter Schultz i Tom Rhodes.

ACPI is a fundamentally new way of discovering devices, managing power usage, and providing standardized access to various hardware previously managed by the BIOS. Progress is being made toward ACPI working on all systems,

but bugs in some motherboards' *ACPI Machine Language* (AML) bytecode, incompleteness in FreeBSD's kernel subsystems, and bugs in the Intel ACPI-CA interpreter continue to appear.

This document is intended to help you assist the FreeBSD ACPI maintainers in identifying the root cause of problems you observe and debugging and developing a solution. Thanks for reading this and we hope we can solve your system's problems.

11.16.1. Submitting Debugging Information

Notatka: Before submitting a problem, be sure you are running the latest BIOS version and, if available, embedded controller firmware version.

For those of you that want to submit a problem right away, please send the following information to freebsd-acpi@FreeBSD.org (<mailto:freebsd-acpi@FreeBSD.org>):

- Description of the buggy behavior, including system type and model and anything that causes the bug to appear. Also, please note as accurately as possible when the bug began occurring if it is new for you.
- The `dmesg(8)` output after `boot -v`, including any error messages generated by you exercising the bug.
- The `dmesg(8)` output from `boot -v` with ACPI disabled, if disabling it helps fix the problem.
- Output from `sysctl hw.acpi`. This is also a good way of figuring out what features your system offers.
- URL where your *ACPI Source Language* (ASL) can be found. Do *not* send the ASL directly to the list as it can be very large. Generate a copy of your ASL by running this command:

```
# acpidump -t -d > name-system.asl
```

(Substitute your login name for *name* and manufacturer/model for *system*. Example: `njl-FooCo6000.asl`)

Most of the developers watch the FreeBSD-CURRENT mailing list

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) but please submit problems to `freebsd-acpi`

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>) to be sure it is seen. Please be patient, all of us have

full-time jobs elsewhere. If your bug is not immediately apparent, we will probably ask you to submit a PR via

`send-pr(1)`. When entering a PR, please include the same information as requested above. This will help us track the problem and resolve it. Do not send a PR without emailing `freebsd-acpi`

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>) first as we use PRs as reminders of existing problems, not a reporting mechanism. It is likely that your problem has been reported by someone before.

11.16.2. Background

ACPI is present in all modern computers that conform to the ia32 (x86), ia64 (Itanium), and amd64 (AMD) architectures. The full standard has many features including CPU performance management, power planes control, thermal zones, various battery systems, embedded controllers, and bus enumeration. Most systems implement less than the full standard. For instance, a desktop system usually only implements the bus enumeration parts while a laptop might have cooling and battery management support as well. Laptops also have suspend and resume, with their own associated complexity.

An ACPI-compliant system has various components. The BIOS and chipset vendors provide various fixed tables (e.g., FADT) in memory that specify things like the APIC map (used for SMP), config registers, and simple

configuration values. Additionally, a table of bytecode (the *Differentiated System Description Table* DSDT) is provided that specifies a tree-like name space of devices and methods.

The ACPI driver must parse the fixed tables, implement an interpreter for the bytecode, and modify device drivers and the kernel to accept information from the ACPI subsystem. For FreeBSD, Intel has provided an interpreter (ACPI-CA) that is shared with Linux and NetBSD. The path to the ACPI-CA source code is `src/sys/contrib/dev/acpica`. The glue code that allows ACPI-CA to work on FreeBSD is in `src/sys/dev/acpica/Osd`. Finally, drivers that implement various ACPI devices are found in `src/sys/dev/acpica`.

11.16.3. Common Problems

For ACPI to work correctly, all the parts have to work correctly. Here are some common problems, in order of frequency of appearance, and some possible workarounds or fixes.

11.16.3.1. Mouse Issues

In some cases, resuming from a suspend operation will cause the mouse to fail. A known work around is to add `hint.psm.0.flags="0x3000"` to the `/boot/loader.conf` file. If this does not work then please consider sending a bug report as described above.

11.16.3.2. Suspend/Resume

ACPI has three suspend to RAM (STR) states, S1-S3, and one suspend to disk state (STD), called S4. S5 is „soft off” and is the normal state your system is in when plugged in but not powered up. S4 can actually be implemented two separate ways. S4BIOS is a BIOS-assisted suspend to disk. S4OS is implemented entirely by the operating system.

Start by checking `sysctl hw.acpi` for the suspend-related items. Here are the results for a Thinkpad:

```
hw.acpi.supported_sleep_state: S3 S4 S5
hw.acpi.s4bios: 0
```

This means that we can use `acpicnf -s` to test S3, S4OS, and S5. If `s4bios` was one (1), we would have S4BIOS support instead of S4 OS.

When testing suspend/resume, start with S1, if supported. This state is most likely to work since it does not require much driver support. No one has implemented S2 but if you have it, it is similar to S1. The next thing to try is S3. This is the deepest STR state and requires a lot of driver support to properly reinitialize your hardware. If you have problems resuming, feel free to email the `freebsd-acpi` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>) list but do not expect the problem to be resolved since there are a lot of drivers/hardware that need more testing and work.

To help isolate the problem, remove as many drivers from your kernel as possible. If it works, you can narrow down which driver is the problem by loading drivers until it fails again. Typically binary drivers like `nvidia.ko`, X11 display drivers, and USB will have the most problems while Ethernet interfaces usually work fine. If you can properly load/unload the drivers, you can automate this by putting the appropriate commands in `/etc/rc.suspend` and `/etc/rc.resume`. There is a commented-out example for unloading and loading a driver. Try setting `hw.acpi.reset_video` to zero (0) if your display is messed up after resume. Try setting longer or shorter values for `hw.acpi.sleep_delay` to see if that helps.

Another thing to try is load a recent Linux distribution with ACPI support and test their suspend/resume support on the same hardware. If it works on Linux, it is likely a FreeBSD driver problem and narrowing down which driver

causes the problems will help us fix the problem. Note that the ACPI maintainers do not usually maintain other drivers (e.g. sound, ATA, etc.) so any work done on tracking down a driver problem should probably eventually be posted to the `freebsd-current` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) list and mailed to the driver maintainer. If you are feeling adventurous, go ahead and start putting some debugging `printf(3)`s in a problematic driver to track down where in its resume function it hangs.

Finally, try disabling ACPI and enabling APM instead. If suspend/resume works with APM, you may be better off sticking with APM, especially on older hardware (pre-2000). It took vendors a while to get ACPI support correct and older hardware is more likely to have BIOS problems with ACPI.

11.16.3.3. System Hangs (temporary or permanent)

Most system hangs are a result of lost interrupts or an interrupt storm. Chipsets have a lot of problems based on how the BIOS configures interrupts before boot, correctness of the APIC (MADT) table, and routing of the *System Control Interrupt* (SCI).

Interrupt storms can be distinguished from lost interrupts by checking the output of `vmstat -i` and looking at the line that has `acpi0`. If the counter is increasing at more than a couple per second, you have an interrupt storm. If the system appears hung, try breaking to DDB (**CTRL+ALT+ESC** on console) and type `show interrupts`.

Your best hope when dealing with interrupt problems is to try disabling APIC support with `hint.apic.0.disabled="1"` in `loader.conf`.

11.16.3.4. Panics

Panics are relatively rare for ACPI and are the top priority to be fixed. The first step is to isolate the steps to reproduce the panic (if possible) and get a backtrace. Follow the advice for enabling `options DDB` and setting up a serial console (see Sekcja 22.6.5.3) or setting up a `dump(8)` partition. You can get a backtrace in DDB with `tr`. If you have to handwrite the backtrace, be sure to at least get the lowest five (5) and top five (5) lines in the trace.

Then, try to isolate the problem by booting with ACPI disabled. If that works, you can isolate the ACPI subsystem by using various values of `debug.acpi.disable`. See the `acpi(4)` manual page for some examples.

11.16.3.5. System Powers Up After Suspend or Shutdown

First, try setting `hw.acpi.disable_on_poweroff="0"` in `loader.conf(5)`. This keeps ACPI from disabling various events during the shutdown process. Some systems need this value set to 1 (the default) for the same reason. This usually fixes the problem of a system powering up spontaneously after a suspend or poweroff.

11.16.3.6. Other Problems

If you have other problems with ACPI (working with a docking station, devices not detected, etc.), please email a description to the mailing list as well; however, some of these issues may be related to unfinished parts of the ACPI subsystem so they might take a while to be implemented. Please be patient and prepared to test patches we may send you.

11.16.4. ASL, acpidump, and IASL

The most common problem is the BIOS vendors providing incorrect (or outright buggy!) bytecode. This is usually manifested by kernel console messages like this:

```
ACPI-1287: *** Error: Method execution failed [\\_SB_.PCI0.LPC0.FIGD._STA] \\
(Node 0xc3f6d160), AE_NOT_FOUND
```

Often, you can resolve these problems by updating your BIOS to the latest revision. Most console messages are harmless but if you have other problems like battery status not working, they are a good place to start looking for problems in the AML. The bytecode, known as AML, is compiled from a source language called ASL. The AML is found in the table known as the DSDT. To get a copy of your ASL, use `acpidump(8)`. You should use both the `-t` (show contents of the fixed tables) and `-d` (disassemble AML to ASL) options. See the [Submitting Debugging Information](#) section for an example syntax.

The simplest first check you can do is to recompile your ASL to check for errors. Warnings can usually be ignored but errors are bugs that will usually prevent ACPI from working correctly. To recompile your ASL, issue the following command:

```
# iasl your.asl
```

11.16.5. Fixing Your ASL

In the long run, our goal is for almost everyone to have ACPI work without any user intervention. At this point, however, we are still developing workarounds for common mistakes made by the BIOS vendors. The Microsoft interpreter (`acpi.sys` and `acpiec.sys`) does not strictly check for adherence to the standard, and thus many BIOS vendors who only test ACPI under Windows never fix their ASL. We hope to continue to identify and document exactly what non-standard behavior is allowed by Microsoft's interpreter and replicate it so FreeBSD can work without forcing users to fix the ASL. As a workaround and to help us identify behavior, you can fix the ASL manually. If this works for you, please send a `diff(1)` of the old and new ASL so we can possibly work around the buggy behavior in ACPI-CA and thus make your fix unnecessary.

Here is a list of common error messages, their cause, and how to fix them:

11.16.5.1. _OS dependencies

Some AML assumes the world consists of various Windows versions. You can tell FreeBSD to claim it is any OS to see if this fixes problems you may have. An easy way to override this is to set `hw.acpi.osname="Windows 2001"` in `/boot/loader.conf` or other similar strings you find in the ASL.

11.16.5.2. Missing Return statements

Some methods do not explicitly return a value as the standard requires. While ACPI-CA does not handle this, FreeBSD has a workaround that allows it to return the value implicitly. You can also add explicit Return statements where required if you know what value should be returned. To force `iasl` to compile the ASL, use the `-f` flag.

11.16.5.3. Overriding the Default AML

After you customize `your.asl`, you will want to compile it, run:

```
# iasl your.asl
```

You can add the `-f` flag to force creation of the AML, even if there are errors during compilation. Remember that some errors (e.g., missing Return statements) are automatically worked around by the interpreter.

DSDT.aml is the default output filename for `iasl`. You can load this instead of your BIOS's buggy copy (which is still present in flash memory) by editing `/boot/loader.conf` as follows:

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/DSDT.aml"
```

Be sure to copy your DSDT.aml to the `/boot` directory.

11.16.6. Getting Debugging Output From ACPI

The ACPI driver has a very flexible debugging facility. It allows you to specify a set of subsystems as well as the level of verbosity. The subsystems you wish to debug are specified as „layers” and are broken down into ACPI-CA components (ACPI_ALL_COMPONENTS) and ACPI hardware support (ACPI_ALL_DRIVERS). The verbosity of debugging output is specified as the „level” and ranges from ACPI_LV_ERROR (just report errors) to ACPI_LV_VERBOSE (everything). The „level” is a bitmask so multiple options can be set at once, separated by spaces. In practice, you will want to use a serial console to log the output if it is so long it flushes the console message buffer. A full list of the individual layers and levels is found in the `acpi(4)` manual page.

Debugging output is not enabled by default. To enable it, add `options ACPI_DEBUG` to your kernel configuration file if ACPI is compiled into the kernel. You can add `ACPI_DEBUG=1` to your `/etc/make.conf` to enable it globally. If it is a module, you can recompile just your `acpi.ko` module as follows:

```
# cd /sys/modules/acpi/acpi
&& make clean &&
make ACPI_DEBUG=1
```

Install `acpi.ko` in `/boot/kernel` and add your desired level and layer to `loader.conf`. This example enables debug messages for all ACPI-CA components and all ACPI hardware drivers (CPU, LID, etc.) It will only output error messages, the least verbose level.

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ERROR"
```

If the information you want is triggered by a specific event (say, a suspend and then resume), you can leave out changes to `loader.conf` and instead use `sysctl` to specify the layer and level after booting and preparing your system for the specific event. The `sysctls` are named the same as the tunables in `loader.conf`.

11.16.7. References

More information about ACPI may be found in the following locations:

- The FreeBSD ACPI mailing list (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>)
- The ACPI Mailing List Archives <http://lists.freebsd.org/pipermail/freebsd-acpi/>

- The old ACPI Mailing List Archives <http://home.jp.FreeBSD.org/mail-list/acpi-jp/>
- The ACPI 2.0 Specification <http://acpi.info/spec.htm>
- FreeBSD Manual pages: `acpi(4)`, `acpi_thermal(4)`, `acpidump(8)`, `iasl(8)`, `acpidb(8)`
- DSDT debugging resource (http://www.cpqlinux.com/acpi-howto.html#fix_broken_dsdt). (Uses Compaq as an example but generally useful.)

Przypisy

1. The auto-tuning algorithm sets `maxusers` equal to the amount of memory in the system, with a minimum of 32, and a maximum of 384.

Rozdział 12. The FreeBSD Booting Process

12.1. Synopsis

The process of starting a computer and loading the operating system is referred to as „the bootstrap process”, or simply „booting”. FreeBSD’s boot process provides a great deal of flexibility in customizing what happens when you start the system, allowing you to select from different operating systems installed on the same computer, or even different versions of the same operating system or installed kernel.

This chapter details the configuration options you can set and how to customize the FreeBSD boot process. This includes everything that happens until the FreeBSD kernel has started, probed for devices, and started `init(8)`. If you are not quite sure when this happens, it occurs when the text color changes from bright white to grey.

After reading this chapter, you will know:

- What the components of the FreeBSD bootstrap system are, and how they interact.
- The options you can give to the components in the FreeBSD bootstrap to control the boot process.
- The basics of `device.hints(5)`.

x86 Only: This chapter only describes the boot process for FreeBSD running on Intel x86 systems.

12.2. The Booting Problem

Turning on a computer and starting the operating system poses an interesting dilemma. By definition, the computer does not know how to do anything until the operating system is started. This includes running programs from the disk. So if the computer can not run a program from the disk without the operating system, and the operating system programs are on the disk, how is the operating system started?

This problem parallels one in the book *The Adventures of Baron Munchausen*. A character had fallen part way down a manhole, and pulled himself out by grabbing his bootstraps, and lifting. In the early days of computing the term *bootstrap* was applied to the mechanism used to load the operating system, which has become shortened to „booting”.

On x86 hardware the Basic Input/Output System (BIOS) is responsible for loading the operating system. To do this, the BIOS looks on the hard disk for the Master Boot Record (MBR), which must be located on a specific place on the disk. The BIOS has enough knowledge to load and run the MBR, and assumes that the MBR can then carry out the rest of the tasks involved in loading the operating system, possibly with the help of the BIOS.

The code within the MBR is usually referred to as a *boot manager*, especially when it interacts with the user. In this case the boot manager usually has more code in the first *track* of the disk or within some OS’s file system. (A boot manager is sometimes also called a *boot loader*, but FreeBSD uses that term for a later stage of booting.) Popular boot managers include **boot0** (a.k.a. **Boot Easy**, the standard FreeBSD boot manager), **Grub**, **GAG**, and **LILO**. (Only **boot0** fits within the MBR.)

If you have only one operating system installed on your disks then a standard PC MBR will suffice. This MBR searches for the first bootable (a.k.a. active) slice on the disk, and then runs the code on that slice to load the

remainder of the operating system. The MBR installed by `fdisk(8)`, by default, is such an MBR. It is based on `/boot/mbr`.

If you have installed multiple operating systems on your disks then you can install a different boot manager, one that can display a list of different operating systems, and allows you to choose the one to boot from. Two of these are discussed in the next subsection.

The remainder of the FreeBSD bootstrap system is divided into three stages. The first stage is run by the MBR, which knows just enough to get the computer into a specific state and run the second stage. The second stage can do a little bit more, before running the third stage. The third stage finishes the task of loading the operating system. The work is split into these three stages because the PC standards put limits on the size of the programs that can be run at stages one and two. Chaining the tasks together allows FreeBSD to provide a more flexible loader.

The kernel is then started and it begins to probe for devices and initialize them for use. Once the kernel boot process is finished, the kernel passes control to the user process `init(8)`, which then makes sure the disks are in a usable state. `init(8)` then starts the user-level resource configuration which mounts file systems, sets up network cards to communicate on the network, and generally starts all the processes that usually are run on a FreeBSD system at startup.

12.3. The Boot Manager and Boot Stages

12.3.1. The Boot Manager

The code in the MBR or boot manager is sometimes referred to as *stage zero* of the boot process. This subsection discusses two of the boot managers previously mentioned: **boot0** and **LILO**.

The boot0 Boot Manager: The MBR installed by FreeBSD's installer or `boot0cfg(8)`, by default, is based on `/boot/boot0`. (The **boot0** program is very simple, since the program in the MBR can only be 446 bytes long because of the slice table and `0x55AA` identifier at the end of the MBR.) If you have installed **boot0** and multiple operating systems on your hard disks, then you will see a display similar to this one at boot time:

Przykład 12-1. boot0 Screenshot

```
F1 DOS
F2 FreeBSD
F3 Linux
F4 ??
F5 Drive 1
```

```
Default: F2
```

Other operating systems, in particular Windows, have been known to overwrite an existing MBR with their own. If this happens to you, or you want to replace your existing MBR with the FreeBSD MBR then use the following command:

```
# fdisk -B -b /boot/boot0 device
```

where *device* is the device that you boot from, such as `ad0` for the first IDE disk, `ad2` for the first IDE disk on a second IDE controller, `da0` for the first SCSI disk, and so on. Or, if you want a custom configuration of the MBR, use `boot0cfg(8)`.

The LILO Boot Manager: To install this boot manager so it will also boot FreeBSD, first start Linux and add the following to your existing `/etc/lilo.conf` configuration file:

```
other=/dev/hdXY
table=/dev/hdX
loader=/boot/chain.b
label=FreeBSD
```

In the above, specify FreeBSD's primary partition and drive using Linux specifiers, replacing *x* with the Linux drive letter and *Y* with the Linux primary partition number. If you are using a SCSI drive, you will need to change `/dev/hd` to read something similar to `/dev/sd`. The `loader=/boot/chain.b` line can be omitted if you have both operating systems on the same drive. Now run `/sbin/lilo -v` to commit your new changes to the system; this should be verified by checking its screen messages.

12.3.2. Stage One, `/boot/boot1`, and Stage Two, `/boot/boot2`

Conceptually the first and second stages are part of the same program, on the same area of the disk. Because of space constraints they have been split into two, but you would always install them together. They are copied from the combined file `/boot/boot` by the installer or **bsdlabel** (see below).

They are located outside file systems, in the first track of the boot slice, starting with the first sector. This is where `boot0`, or any other boot manager, expects to find a program to run which will continue the boot process. The number of sectors used is easily determined from the size of `/boot/boot`.

`boot1` is very simple, since it can only be 512 bytes in size, and knows just enough about the FreeBSD *bsdlabel*, which stores information about the slice, to find and execute `boot2`.

`boot2` is slightly more sophisticated, and understands the FreeBSD file system enough to find files on it, and can provide a simple interface to choose the kernel or loader to run.

Since the loader is much more sophisticated, and provides a nice easy-to-use boot configuration, `boot2` usually runs it, but previously it was tasked to run the kernel directly.

Przykład 12-2. `boot2` Screenshot

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

If you ever need to replace the installed `boot1` and `boot2` use `bsdlabel(8)`:

```
# bsdlabel -B diskslice
```

where *diskslice* is the disk and slice you boot from, such as `ad0s1` for the first slice on the first IDE disk.

Dangerously Dedicated Model If you use just the disk name, such as `ad0`, in the `bsdlabel(8)` command you will create a dangerously dedicated disk, without slices. This is almost certainly not what you want to do, so make sure you double check the `bsdlabel(8)` command before you press **Return**.

12.3.3. Stage Three, `/boot/loader`

The loader is the final stage of the three-stage bootstrap, and is located on the file system, usually as `/boot/loader`.

The loader is intended as a user-friendly method for configuration, using an easy-to-use built-in command set, backed up by a more powerful interpreter, with a more complex command set.

12.3.3.1. Loader Program Flow

During initialization, the loader will probe for a console and for disks, and figure out what disk it is booting from. It will set variables accordingly, and an interpreter is started where user commands can be passed from a script or interactively.

The loader will then read `/boot/loader.rc`, which by default reads in `/boot/defaults/loader.conf` which sets reasonable defaults for variables and reads `/boot/loader.conf` for local changes to those variables. `loader.rc` then acts on these variables, loading whichever modules and kernel are selected.

Finally, by default, the loader issues a 10 second wait for key presses, and boots the kernel if it is not interrupted. If interrupted, the user is presented with a prompt which understands the easy-to-use command set, where the user may adjust variables, unload all modules, load modules, and then finally boot or reboot.

12.3.3.2. Loader Built-In Commands

These are the most commonly used loader commands. For a complete discussion of all available commands, please see `loader(8)`.

`autoboot seconds`

Proceeds to boot the kernel if not interrupted within the time span given, in seconds. It displays a countdown, and the default time span is 10 seconds.

`boot [-options] [kernelname]`

Immediately proceeds to boot the kernel, with the given options, if any, and with the kernel name given, if it is.

`boot-conf`

Goes through the same automatic configuration of modules based on variables as what happens at boot. This only makes sense if you use `unload` first, and change some variables, most commonly `kernel`.

`help [topic]`

Shows help messages read from `/boot/loader.help`. If the topic given is `index`, then the list of available topics is given.

`include filename ...`

Processes the file with the given filename. The file is read in, and interpreted line by line. An error immediately stops the include command.

`load [-t type] filename`

Loads the kernel, kernel module, or file of the type given, with the filename given. Any arguments after filename are passed to the file.

`ls [-l] [path]`

Displays a listing of files in the given path, or the root directory, if the path is not specified. If `-l` is specified, file sizes will be shown too.

`lsdev [-v]`

Lists all of the devices from which it may be possible to load modules. If `-v` is specified, more details are printed.

`lsmod [-v]`

Displays loaded modules. If `-v` is specified, more details are shown.

`more filename`

Displays the files specified, with a pause at each `LINES` displayed.

`reboot`

Immediately reboots the system.

`set variable`

`set variable=value`

Sets the loader's environment variables.

`unload`

Removes all loaded modules.

12.3.3.3. Loader Examples

Here are some practical examples of loader usage:

- To simply boot your usual kernel, but in single-user mode:

```
boot -s
```

- To unload your usual kernel and modules, and then load just your old (or another) kernel:

```
unload
load kernel.old
```

You can use `kernel.GENERIC` to refer to the generic kernel that comes on the install disk, or `kernel.old` to refer to your previously installed kernel (when you have upgraded or configured your own kernel, for example).

Notatka: Use the following to load your usual modules with another kernel:

```
unload
set kernel="kernel.old"
boot-conf
```

- To load a kernel configuration script (an automated script which does the things you would normally do in the kernel boot-time configurator):

```
load -t userconfig_script /boot/kernel.conf
```

12.4. Kernel Interaction During Boot

Once the kernel is loaded by either loader (as usual) or boot2 (bypassing the loader), it examines its boot flags, if any, and adjusts its behavior as necessary.

12.4.1. Kernel Boot Flags

Here are the more common boot flags:

- a
during kernel initialization, ask for the device to mount as the root file system.
- C
boot from CDROM.
- c
run UserConfig, the boot-time kernel configurator
- s
boot into single-user mode
- v
be more verbose during kernel startup

Notatka: There are other boot flags, read boot(8) for more information on them.

12.5. Device Hints

Contributed by Tom Rhodes.

Notatka: This is a FreeBSD 5.0 and later feature which does not exist in earlier versions.

During initial system startup, the boot loader(8) will read the device.hints(5) file. This file stores kernel boot information known as variables, sometimes referred to as „device hints”. These „device hints” are used by device drivers for device configuration.

Device hints may also be specified at the Stage 3 boot loader prompt. Variables can be added using `set`, removed with `unset`, and viewed with the `show` commands. Variables set in the `/boot/device.hints` file can be

overridden here also. Device hints entered at the boot loader are not permanent and will be forgotten on the next reboot.

Once the system is booted, the `kenv(1)` command can be used to dump all of the variables.

The syntax for the `/boot/device.hints` file is one variable per line, using the standard hash `„#”` as comment markers. Lines are constructed as follows:

```
hint.driver.unit.keyword="value"
```

The syntax for the Stage 3 boot loader is:

```
set hint.driver.unit.keyword=value
```

`driver` is the device driver name, `unit` is the device driver unit number, and `keyword` is the hint keyword. The keyword may consist of the following options:

- `at`: specifies the bus which the device is attached to.
- `port`: specifies the start address of the I/O to be used.
- `irq`: specifies the interrupt request number to be used.
- `drq`: specifies the DMA channel number.
- `maddr`: specifies the physical memory address occupied by the device.
- `flags`: sets various flag bits for the device.
- `disabled`: if set to 1 the device is disabled.

Device drivers may accept (or require) more hints not listed here, viewing their manual page is recommended. For more information, consult the `device.hints(5)`, `kenv(1)`, `loader.conf(5)`, and `loader(8)` manual pages.

12.6. Init: Process Control Initialization

Once the kernel has finished booting, it passes control to the user process `init(8)`, which is located at `/sbin/init`, or the program path specified in the `init_path` variable in `loader`.

12.6.1. Automatic Reboot Sequence

The automatic reboot sequence makes sure that the file systems available on the system are consistent. If they are not, and `fsck(8)` cannot fix the inconsistencies, `init(8)` drops the system into single-user mode for the system administrator to take care of the problems directly.

12.6.2. Single-User Mode

This mode can be reached through the automatic reboot sequence, or by the user booting with the `-s` option or setting the `boot_single` variable in `loader`.

It can also be reached by calling `shutdown(8)` without the `reboot (-r)` or `halt (-h)` options, from multi-user mode.

If the system console is set to `insecure` in `/etc/ttys`, then the system prompts for the `root` password before initiating single-user mode.

Przykład 12-3. An Insecure Console in `/etc/ttys`

```
# name  getty                                type    status      comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                                unknown off insecure
```

Notatka: An `insecure` console means that you consider your physical security to the console to be insecure, and want to make sure only someone who knows the `root` password may use single-user mode, and it does not mean that you want to run your console insecurely. Thus, if you want security, choose `insecure`, not `secure`.

12.6.3. Multi-User Mode

If `init(8)` finds your file systems to be in order, or once the user has finished in single-user mode, the system enters multi-user mode, in which it starts the resource configuration of the system.

12.6.3.1. Resource Configuration (`rc`)

The resource configuration system reads in configuration defaults from `/etc/defaults/rc.conf`, and system-specific details from `/etc/rc.conf`, and then proceeds to mount the system file systems mentioned in `/etc/fstab`, start up networking services, start up miscellaneous system daemons, and finally runs the startup scripts of locally installed packages.

The `rc(8)` manual page is a good reference to the resource configuration system, as is examining the scripts themselves.

12.7. Shutdown Sequence

Upon controlled shutdown, via `shutdown(8)`, `init(8)` will attempt to run the script `/etc/rc.shutdown`, and then proceed to send all processes the `TERM` signal, and subsequently the `KILL` signal to any that do not terminate timely.

To power down a FreeBSD machine on architectures and systems that support power management, simply use the command `shutdown -p now` to turn the power off immediately. To just reboot a FreeBSD system, just use `shutdown -r now`. You need to be `root` or a member of `operator` group to run `shutdown(8)`. The `halt(8)` and `reboot(8)` commands can also be used, please refer to their manual pages and to `shutdown(8)`'s one for more information.

Notatka: Power management requires `acpi(4)` support in the kernel or loaded as module for.

Rozdział 13. Użytkownicy i podstawy zarządzania kontami

Napisał Neil Blakey-Milner. Przetłumaczył Cezary Morgia.

13.1. Streszczenie

FreeBSD umożliwia pracę z komputerem wielu użytkownikom w tym samym czasie. Oczywiście, tylko jedna osoba może w danej chwili siedzieć przed monitorem i klawiaturą¹, lecz dowolna liczba użytkowników może być zalogowana przez sieć. By móc korzystać z systemu, każdy użytkownik musi posiadać konto.

Po przeczytaniu tego rozdziału, będziemy wiedzieć:

- Jakie są różnice pomiędzy różnymi kontami użytkowników w systemie FreeBSD.
- Jak dodawać konta użytkowników.
- Jak usuwać konta użytkowników.
- Jak modyfikować szczegóły konta, np. pełną nazwę użytkownika czy preferowaną powłokę.
- Jak ograniczać dostęp do zasobów takich jak pamięć i czas procesora w oparciu o konta i grupy użytkowników.
- Jak korzystać z grup by ułatwić zarządzanie kontami.

Przed przeczytaniem tego rozdziału, powinniśmy:

- Znać podstawy obsługi systemów UNIX i FreeBSD (Rozdział 3).

13.2. Wprowadzenie

Konta umożliwiają dostęp do systemu. Natomiast procesy uruchamiane są przez użytkowników. Stąd też zarządzanie kontami systemowymi i użytkowników stanowi integralny element systemów FreeBSD.

Z każdym kontem w systemie FreeBSD skojarzone są pewne informacje służące do jego identyfikacji.

Nazwa użytkownika

Nazwa użytkownika, jaka byłaby podana w zgłoszeniu `login:`. Nazwy użytkowników muszą być unikalne w danym komputerze; nie możemy posiadać dwóch użytkowników o tej samej nazwie. Kilka reguł tworzenia poprawnych nazw użytkowników opisuje podręcznik `passwd(5)`. Z reguły stosuje się nazwy składające się z ośmiu bądź mniejszej ilości znaków pisanych małą literą.

Hasło

Każde konto posiada hasło. Hasło może być puste, w tej sytuacji nie będzie wymagane żadne hasło by uzyskać dostęp do systemu. Z reguły jest to bardzo złym pomysłem; każde konto powinno zabezpieczone hasłem.

Identyfikator użytkownika (UID)

UID jest numerem, tradycyjnie z przedziału od 0 do 65535², wykorzystywanym do jednoznacznej identyfikacji użytkownika w systemie. Do identyfikacji użytkowników FreeBSD wewnętrznie wykorzystuje UID — każde polecenie FreeBSD przyjmujące nazwę użytkownika jako argument zamieni ją na identyfikator UID przed rozpoczęciem. Oznacza to, że możemy posiadać kilka kont o różnych nazwach użytkowników ale o tym samym UID. Z punktu widzenia FreeBSD konta te są jednym i tym samym użytkownikiem. Tym nie mniej jest mało prawdopodobne byśmy kiedykolwiek tego potrzebowali.

Identyfikator grupy (GID)

GID jest numerem, tradycyjnie z przedziału od 0 do 65535², wykorzystywanym do jednoznacznej identyfikacji podstawowej grupy, do której należy użytkownik. Grupy stanowią mechanizm kontroli dostępu do zasobów na podstawie identyfikatora GID użytkownika w miejsce jego UID. Pozwala to znacznie zredukować rozmiar niektórych plików konfiguracyjnych. Użytkownik może należeć do więcej niż jednej grupy.

Klasa logowania

Klasy logowania stanowią rozszerzenie mechanizmu grup, dostarczając dodatkowej elastyczności w procesie dopasowywania systemu do różnych użytkowników.

Czas zmiany hasła

Domyślnie FreeBSD nie zmusza użytkowników do regularnego zmieniania haseł. Możemy jednak wymusić takie zachowanie narzucając niektórym bądź wszystkim użytkownikom zmianę ich haseł w określonych odstępach czasowych.

Czas wygaśnięcia konta

Domyślnie FreeBSD nie stosuje wygaszania kont. Jeśli tworzymy konta, o których wiemy, że będą miały ograniczoną żywotność, np. w szkole, gdzie zarządzamy kontami studentów, możemy określić kiedy konto wygaśnie. Po upływie terminu wygaśnięcia konto nie będzie mogło być wykorzystane do logowania się do systemu, chociaż katalogi i pliki użytkownika pozostaną w systemie.

Pełna nazwa użytkownika

Nazwa użytkownika w sposób jednoznaczny identyfikuje konta we FreeBSD, jednakże dość często nie odzwierciedla prawdziwych nazwisk użytkowników. Informacja ta może zostać dołączona do konta.

Katalog macierzysty

Katalog macierzysty określa pełną ścieżkę dostępu do katalogu w systemie, w którym dany użytkownik będzie rozpoczynał pracę po każdym zalogowaniu się do systemu. Z reguły katalogi macierzyste wszystkich użytkowników umieszczane są w katalogach `/home/nazwa_użytkownika` lub `/usr/home/nazwa_użytkownika`. W swoich katalogach macierzystych użytkownicy przechowują pliki i katalogi, które mogą tworzyć, modyfikować i usuwać według własnego uznania.

Powłoka użytkownika

Powłoka udostępnia użytkownikowi domyślne środowisko wykorzystywane do interakcji z systemem. Istnieje wiele różnych rodzajów powłok. Doświadczeni użytkownicy z pewnością posiadają własne upodobania, które mogą mieć odwzorowanie w ustawieniach ich kont.

Istnieją trzy podstawowe typy kont: konto Superużytkownika, konta systemowe i konta użytkowników. Konto Superużytkownika, z reguły nazywane `root`, wykorzystywane jest do zarządzania systemem z nieograniczonymi

uprawnieniami. Konta systemowe wykorzystywane są do uruchamiania usług. Konta użytkowników natomiast wykorzystywane są przez prawdziwych ludzi, którzy logują się, czytają pocztę itp.

13.3. Konto superużytkownika

Konto superużytkownika, z reguły nazywane kontem `root`, domyślnie jest tak skonfigurowane, by ułatwić administrację systemem i jako takie nie powinno być wykorzystywane do codziennych zajęć, jak np. wysyłanie i odbiór poczty, ogólne zgłębianie systemu czy programowanie.

Powodem tego jest fakt, iż superużytkownik - w przeciwieństwie do zwykłego użytkownika - może działać bez żadnych ograniczeń. Stąd też niewłaściwe użycie tego konta może skończyć się spektakularną katastrofą. Konta użytkowników nie są zdolne przez pomyłkę uszkodzić systemu. Tak więc najlepszym rozwiązaniem jest korzystanie z konta zwykłego użytkownika kiedy tylko to możliwe, chyba że szczególnie potrzebujemy dodatkowych uprawnień.

Skoro dodatkowa spacja bądź brak znaku mogą doprowadzić do nieodwracalnej utraty danych, zawsze powinniśmy sprawdzać polecenie wydawane jako superużytkownik dwu- lub nawet trzykrotnie.

Tak więc pierwszą rzeczą, którą powinniśmy zrobić po przeczytaniu tego rozdziału - jeśli jeszcze tego nie zrobiliśmy - jest stworzenie dla siebie konta nieuprzywilejowanego użytkownika do ogólnego zastosowania. Dotyczy to zarówno sytuacji gdy korzystamy z systemu o wielu użytkownikach jak i gdy pracujemy na maszynie sami. W dalszej części tego rozdziału omówiony zostanie proces tworzenia dodatkowego konta oraz zmiany trybu pracy pomiędzy zwykłym użytkownikiem a superużytkownikiem.

13.4. Konta systemowe

Konta systemowe wykorzystywane są do uruchamiania usług, takich jak np. DNS, poczta, serwery WWW itd. Powodem takiego wykorzystania kont jest bezpieczeństwo; jeśli wszystkie usługi pracowałyby jako superużytkownik, mogłyby działać bez żadnych ograniczeń.

Przykładami kont systemowych są `daemon`, `operator`, `bind` (dla serwera DNS), `news` oraz `www`.

`nobody` jest standardowym nieuprzywilejowanym kontem systemowym. Tym nie mniej należy pamiętać, że im więcej usług będzie pracowało jako użytkownik `nobody`, z tym większą ilością plików i procesów będzie on związany i tym samym uzyska większe przywileje.

13.5. Konta użytkowników

Konta użytkowników są dla rzeczywistych osób podstawowym środkiem dostępu do systemu. Izolują one użytkowników od otoczenia, uniemożliwiając im uszkodzenie systemu bądź danych innych użytkowników, pozwalając im przy tym dostosować do własnych potrzeb środowisko pracy.

Każda osoba korzystająca z naszego systemu powinna posiadać unikalne konto użytkownika. Pozwala to nam sprawdzić co kto robi, a także uniemożliwić użytkownikom ingerencję w ustawienia innych użytkowników, czytać sobie na wzajem pocztę itd.

Każdy użytkownik może dostosować własne środowisko pracy wykorzystując alternatywne powłoki, edytory, skróty klawiszowe czy język.

13.6. Modyfikacja ustawień kont

W środowisku systemów UNIX dostępnych jest szeroka gama różnorodnych poleceń do manipulacji kontami użytkowników. Najczęściej wykorzystywane zostały omówione poniżej, wraz z przykładami zastosowania.

Polecenie	Opis
<code>adduser(8)</code>	Zalecane pogram wiersza poleceń do dodawania nowych użytkowników.
<code>rmuser(8)</code>	Zalecany program wiersza poleceń do usuwania użytkowników.
<code>chpass(1)</code>	Elastyczne narzędzie do zmiany informacji w bazie danych użytkowników.
<code>passwd(1)</code>	Proste polecenie wiersza poleceń do zmiany haseł użytkowników.
<code>pw(8)</code>	Potężne i elastyczne narzędzie do zmiany wszystkich parametrów kont użytkowników.

13.6.1. `adduser`

`adduser(8)` jest prostym programem służącym do dodawania nowych użytkowników. Tworzy ono odpowiednie wpisy w plikach systemowych `passwd` i `group`, tworzy katalog macierzysty nowego użytkownika oraz kopiuje z `/usr/share/skel` domyślne pliki konfiguracyjne („dotfiles”). Potrafi również wysłać nowemu użytkownikowi wiadomość powitalną.

Notatka: Przy wprowadzaniu hasła na ekranie nie są wyświetlane żadne znaki, nawet gwiazdki. Dlatego właśnie w tym momencie należy zwrócić szczególną uwagę, by się nie pomylić.

Przykład 13-1. Dodawanie użytkownika we FreeBSD

```
# adduser
Username: jru
Full name: J. Random User
Uid (Leave empty for default):
Login group [jru]:
Login group is jru. Invite jru into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh zsh nologin) [sh]: zsh
Home directory [/home/jru]:
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password:
Enter password again:
Lock out the account after creation? [no]:
Username      : jru
Password      : ****
Full Name     : J. Random User
Uid           : 1001
Class         :
Groups        : jru wheel
Home          : /home/jru
Shell         : /usr/local/bin/zsh
```

```

Locked      : no
OK? (yes/no): yes
adduser: INFO: Successfully added (jru) to the user database.
Add another user? (yes/no): no
Goodbye!
#

```

13.6.2. rmuser

By całkowicie usunąć użytkownika z systemu, możemy skorzystać z `rmuser(8)`. Narzędzie to wykonuje następujące czynności:

1. Usuwa wpisy użytkownika w `crontab(1)` (jeśli jakieś są).
2. Usuwa wszystkie prace `at(1)` należące do użytkownika.
3. Niszczy wszystkie procesy użytkownika.
4. Usuwa użytkownika z lokalnego pliku haseł w systemie.
5. Usuwa katalog macierzysty użytkownika (jeśli jego właścicielem jest dany użytkownik).
6. Usuwa pocztę należącą do użytkownika z `/var/mail`.
7. Usuwa wszystkie pliki z systemów tymczasowych, np. `/tmp`, których właścicielem jest użytkownik.
8. Ostatecznie, usuwa użytkownika z wszystkich grup w `/etc/group`, do których należy.

Notatka: Jeśli po usunięciu użytkownika grupa pozostanie pusta a nazwa tej grupy jest taka sama jak nazwa użytkownika, grupa jest również usuwana; dotyczy to przede wszystkim grup użytkowników stworzonych przez `adduser(8)`.

Nie można wykorzystać `rmuser(8)` do usunięcia konta superużytkownika, gdyż prawie zawsze wskazuje to na masową destrukcję.

Domyślnie wykorzystywany jest tryb interaktywny, który stara upewnić się, że wiemy co robimy.

Przykład 13-2. Tryb interaktywny `rmuser`

```

# rmuser jru
Matching password entry:
jru:*:1001:1001::0:0:J. Random User:/home/jru:/usr/local/bin/zsh
Is this the entry you wish to remove? y
Remove user's home directory (/home/jru)? y
Updating password file, updating databases, done.
Updating group file: trusted (removing group jru -- personal group is empty) done.
Removing user's incoming mail file /var/mail/jru: done.
Removing files belonging to jru from /tmp: done.
Removing files belonging to jru from /var/tmp: done.
Removing files belonging to jru from /var/tmp/vi.recover: done.
#

```

13.6.3. chpass

chpass(1) zmienia w bazie danych użytkowników parametry konta, jak np. hasło, powłokę czy inne szczegółowe informacje.

Jedynie administrator systemu, jako superużytkownik, może zmieniać parametry kont i hasła innych użytkowników za pomocą chpass(1).

Jeśli nie podamy żadnych opcji bądź jedynie nazwę użytkownika, chpass(1) uruchomi edytor informacji o użytkowniku. Po wyjściu z edytora, baza danych użytkowników zostanie aktualizowana.

Notatka: Jeśli nie jesteśmy superużytkownikiem, przed opuszczeniem edytora zostaniemy zapytani o hasło.

Przykład 13-3. Tryb interaktywny chpass superużytkownika

```
#Changing user database information for jru.
Login: jru
Password: *
Uid [#]: 1001
Gid [# or name]: 1001
Change [month day year]:
Expire [month day year]:
Class:
Home directory: /home/jru
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

Zwykli użytkownicy mogą zmienić jedynie część tych informacji i jedynie własnych kont.

Przykład 13-4. Tryb interaktywny chpass zwykłego użytkownika

```
#Changing user database information for jru.
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

Notatka: chfn(1) i chsh(1) są jedynie dowiązaniem do chpass(1), podobnie jak ypchpass(1), ypchfn(1) i ypchsh(1). Obsługa NIS jest automatyczna, tak więc dopisywanie yp przed poleceniem nie jest potrzebne. Jeśli jest to dla nas niezrozumiałe, nie martwmy się, Rozdział 25 opisuje NIS szczegółowo.

13.6.4. passwd

Użycie polecenia `passwd(1)` jest typowym sposobem zmiany własnego hasła, bądź hasła innego użytkownika jako superużytkownik.

Notatka: By uniknąć przypadkowych bądź nieuprawnionych zmian, nim będziemy mogli podać nowe hasło, musimy wpierw wpisać dotychczasowe.

Przykład 13-5. Zmiana własnego hasła

```
% passwd
Changing local password for jru.
Old password:
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Przykład 13-6. Zmiana hasła innego użytkownika jako superużytkownik

```
# passwd jru
Changing local password for jru.
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Notatka: Podobnie jak w przypadku `chpasswd(1)`, `yppasswd(1)` jest jedynie dowiązaniem do `passwd(1)`. Tak więc NIS działa poprawnie z obydwoma poleceniami.

13.6.5. pw

`pw(8)` jest programem wiersza poleceń służącym do tworzenia, usuwania, modyfikowania i wyświetlania użytkowników i grup. Działa jako pośrednik do systemowych plików użytkowników i grup. `pw(8)` dysponuje bardzo potężnym zestawem opcji, które umożliwiają wykorzystanie go w skryptach powłoki. Jednakże, nowym użytkownikom może wydać się zbyt skomplikowany w porównaniu z innymi przedstawionymi tu poleceniami.

13.7. Ograniczanie użytkowników

Jeśli w naszym systemie mamy wielu użytkowników konieczne może się okazać ograniczenie ich możliwości. FreeBSD posiada kilka sposobów na ograniczenie ilości zasobów systemowych, które dana osoba może wykorzystać. Ograniczenia te są podzielone na dwie grupy: udziały dyskowe i limity innych zasobów.

Udziały dyskowe (ang. quota) ograniczają dostępną użytkownikowi przestrzeń na dysku. Umożliwiają również szybkie sprawdzenie stopnia jej wykorzystania bez konieczności każdorazowego przeliczania. Udziały dyskowe szczegółowo omawia Sekcja 17.15.

Pozostałe limity dają możliwość ograniczenia wykorzystania czasu procesora, pamięci i innych zasobów, z których może korzystać użytkownik. Definiuje się je przy wykorzystaniu klas logowania i to właśnie one zostały tutaj opisane.

Klasy logowania określone są w pliku `/etc/login.conf`. Szczegółowa semantyka wykracza dalece poza ramy tej sekcji. Jej pełen opis znaleźć można w podręczniku systemowym `login.conf(5)`. W tym momencie wystarczy powiedzieć, że każdy użytkownik posiada przypisaną klasę logowania (domyślnie jest to klasa `default`) przy czym każda klasa logowania dysponuje zestawem uprawnień. Uprawnieniami logowania są pary *nazwa=wartość*, gdzie *nazwa* jest identyfikatorem a *wartość* dowolnym ciągiem przetwarzanym w sposób zależny od nazwy.

Konfiguracja klas logowania i uprawnień jest stosunkowo prostym zadaniem i również została opisana na stronach podręcznika systemowego `login.conf(5)`.

Notatka: Normalnie system nie odczytuje konfiguracji bezpośrednio z pliku `/etc/login.conf`, lecz odczytuje plik bazy danych `/etc/login.conf.db`, który umożliwia szybsze przeszukiwanie. By wygenerować plik `/etc/login.conf.db` z `/etc/login.conf` należy uruchomić następujące polecenie:

```
# cap_mkdb /etc/login.conf
```

Limity zasobów różnią się od typowych uprawnień logowania. Po pierwsze, dla każdego limitu istnieje tzw. miękki (aktualny) i twardy limit. Limit miękki może być zmieniany przez użytkownika bądź aplikację, lecz nie może być większy od twardego. Twardy natomiast może być zmniejszony przez użytkownika, lecz nie podwyższony. Po drugie, większość limitów zasobów dotyczy procesów danego użytkownika, nie użytkownika jako całości. Przy czym zauważyć należy, że różnice te wynikają ze względu na odmienne metody zarządzania limitami, nie zaś na implementację struktury uprawnień logowania (np. nie są one *na prawdę* specjalnym przypadkiem uprawnień logowania).

Zatem, bez dalszych wstępów, zestawione poniżej zostały najczęściej wykorzystywane limity zasobów (opis pozostałych, razem z innymi uprawnieniami logowania, odnaleźć można w podręczniku `login.conf(5)`).

`coredumpsize`

Ograniczenie rozmiaru pliku rdzenia generowanego przez program. Jest ono - z oczywistych powodów - podrzędne w stosunku do innych limitów wykorzystania przestrzeni dysku (np. `filesize` bądź udziałów dyskowych). Niemniej, limit ten jest powszechnie stosowany jako mniej surowa metoda kontroli wykorzystania przestrzeni na dysku: skoro użytkownicy sami nie generują plików rdzenia i często ich nie usuwają, ustawienie tego limitu może uchronić ich od utraty wolnej przestrzeni na dysku gdy duży program (np. **emacs**) ulegnie awarii.

`cputime`

Jest to maksymalny czas procesora jaki może pochłaniać proces użytkownika. Procesy przekraczające zostaną zniszczone przez jądro.

Notatka: Jest to ograniczenie ilości wykorzystywanego czasu procesora, nie procent wykorzystania jaki wyświetlany jest w niektórych polach `top(1)` i `ps(1)`. Ograniczenie tego drugiego, na chwilę pisanie tego tekstu, nie jest możliwe i byłoby raczej bezużyteczne: przykładowo kompilator — pracujący jako prawdopodobnie uzasadnione zadanie — może bardzo łatwo przez pewien czas zająć prawie 100% procesora.

`filesize`

Jest to maksymalny rozmiar pliku, który może posiadać użytkownik. W przeciwieństwie do udziałów dyskowych, limit ten rzutuje na wielkość poszczególnych plików, których właścicielem jest użytkownik, nie zaś na wszystkie jako całość.

`maxproc`

Jest to maksymalna liczba procesów, które użytkownik może uruchomić. Dotyczy to zarówno procesy pierwszoplanowe jak i pracujące w tle. Z oczywistych względów, wartość ta nie może być większa niż ograniczenia systemowe parametru `sysctl(8)` `kern.maxproc`. Warto również zwrócić uwagę, iż zbyt niska wartość może utrudnić codzienną pracę: dość często okazuje się być przydatną możliwość zalogowania się w wielu sesjach bądź wykonywania poleceń potokowo. Niektóre zadania, jak np. kompilacja dużego programu, również tworzą wiele procesów (np. `make(1)`, `cc(1)` i inne preprocesory pośrednie).

`memorylocked`

Jest to maksymalna wielkość pamięci jakiej proces użytkownika może zablokować w pamięci głównej (np. patrz `mlock(2)`). Niektóre bardzo istotne dla poprawnego funkcjonowania systemu programy, jak np. `amd(8)`, blokują się w głównej pamięci w taki sposób, że w przypadku problemów nie przykładają się do zniszczenia systemu.

`memoryuse`

Jest to maksymalna wielkość pamięci, którą proces może w danym czasie wykorzystać. Dotyczy to zarówno pamięci głównej jak i przestrzeni wymiany. Limit ten nie wyłącza wszystkich potencjalnych sytuacji wymagających ograniczania wykorzystania pamięci, lecz stanowi dobry początek.

`openfiles`

Jest to maksymalna liczba plików, które proces może otworzyć. We FreeBSD pliki wykorzystywane są również do reprezentacji gniazd i kanałów IPC. Tym samym należy zachować ostrożność przy doborze właściwej wartości, by nie ustawić jej na zbyt niską. Ogólne ograniczenie tej wartości dla całego systemu definiowane jest przez zmienną `sysctl(8)` `kern.maxfiles`.

`sbsize`

Jest to ograniczenie pamięci sieciowej, w tym również mbufs, którą może wykorzystać użytkownik. Limit ten został wprowadzony jako odpowiedź na stary atak DoS polegający na stworzeniu wielu gniazd sieciowych, lecz może również być wykorzystywany jako metoda ograniczenia komunikacji sieciowej.

`stacksize`

Jest to maksymalny rozmiar, do którego może urosnąć stos procesu. w pamięci. Sam z siebie nie jest on wystarczający by ograniczyć ilość pamięci wykorzystywanej przez program. Powinien być stosowany w parze z innymi limitami.

Jest jeszcze kilka innych kwestii wartych pamiętania przy konfiguracji limitów zasobów. Poniżej zamieszczonych jest kilka ogólnych porad, sugestii i rozmaitych komentarzy.

- Procesy uruchomione w trakcie uruchamiania systemu przez `/etc/rc` są przypisane do klasy logowania `daemon`.
- Chociaż plik `/etc/login.conf` dostarczany wraz z systemem jest dobrym źródłem rozsądnych wartości dla większości ograniczeń, jedynie my - administratorzy - możemy wiedzieć, co jest odpowiednie dla naszego systemu. Ustawienie zbyt wysokich limitów może otworzyć nasz system na nadużycia, podczas gdy ustawienie za niskich może utrudnić codzienną pracę.
- Użytkownikom Systemu okien X (X11) powinno prawdopodobnie przyznać się więcej zasobów niż innym użytkownikom. X11 samo z siebie wykorzystuje dużo zasobów systemowych, lecz również zachęca użytkowników do jednoczesnego uruchamiania większej ilości programów.
- Pamiętajmy, że wiele limitów ma zastosowanie do indywidualnych procesów, nie użytkownika jako całości. Przykładowo, ustawienie `openfiles` na 50 oznacza, że każdy proces użytkownika może otworzyć do 50 plików. Tym samym całkowita liczba plików, które może otworzyć użytkownik wynosi wartość `openfiles` pomnożona o wartość `maxproc`. To samo dotyczy konsumpcji pamięci.

Więcej informacji o limitach zasobów, klasach logowania i ogólnych uprawnieniach znaleźć można w odpowiednich podręcznikach systemowych: `cap_mkdb(1)`, `getrlimit(2)`, `login.conf(5)`.

13.8. Grupy

Grupa jest po prostu listą użytkowników. Grupy identyfikowane są na podstawie nazwy grupy oraz numeru GID (ang. Group ID). We FreeBSD (i w większości innych systemów UNIX), dwoma czynnikami wykorzystywanymi przez jądro do określenia czy dany proces jest uprawniony do danej czynności jest identyfikator użytkownika (UID) oraz lista grup, do których użytkownik należy. W przeciwieństwie do UID, proces dysponuje listą grup powiązanych z nim. W różnych źródłach możemy przeczytać, że pewne rzeczy odnoszą się do „identyfikatora grupy” użytkownika bądź procesu; w większości przypadków oznacza to po prostu pierwszą grupę na liście.

Odwzorowanie nazwy grupy na identyfikator grupy znajduje się w pliku `/etc/group`. Jest to prosty plik tekstowy z czterema polami oddzielonymi dwukropkiem. Pierwsze pole zawiera nazwę grupy, drugie zaszyfrowane hasło, trzecie identyfikator grupy a czwarte listę członków grupy oddzielonych przecinkami. Plik ten można śmiało edytować ręcznie (zakładając oczywiście, że nie popełnimy żadnych błędów składniowych!). Szczegółowy opis składni pliku znaleźć można na stronie podręcznika systemowego `group(5)`.

Jeśli nie chcemy ręcznie edytować pliku `/etc/group`, by dodawać i usuwać grupy, możemy wykorzystać polecenie `pw(8)`. Przykładowo, by dodać grupę o nazwie `teamtwo` a następnie potwierdzić, że została dodana możemy użyć:

Przykład 13-7. Dodawanie grupy za pomocą pw(8)

```
# pw groupadd teamtwo
# pw groupshow teamtwo
teamtwo:*:1100:
```

Numer 1100 powyżej jest identyfikatorem grupy `teamtwo`. W chwili obecnej grupa ta nie zawiera żadnych członków i tym samym jest raczej bezużyteczna. Zmieńmy to zapraszając do grupy użytkownika `jru`.

Przykład 13-8. Dodawanie użytkownika do grupy za pomocą pw(8)

```
# pw groupmod teamtwo -M jru
# pw groupshow teamtwo
teamtwo:*:1100:jru
```

Parametrem opcji `-M` jest lista użytkowników oddzielonych przecinkami, którzy są członkami grupy. Z poprzednich sekcji wiemy, że plik haseł również zawiera przypisaną grupę dla każdego użytkownika. Jest to grupa, do której użytkownik został automatycznie dodany przez system. Taki użytkownik nie zostanie wyświetlony jako członek grupy gdy wykorzystamy polecenie `pw(8) groupshow`, jednakże wciąż będzie wyświetlany przy użyciu polecenia `id(1)` bądź innego podobnego narzędzia. Innymi słowy, `pw(8)` manipuluje jedynie plikiem `/etc/group` i nigdy nie próbuje odczytać dodatkowych informacji z pliku `/etc/passwd`.

Przykład 13-9. Wykorzystanie id(1) do określenia członkostwa w grupach

```
% id jru
uid=1001(jru) gid=1001(jru) groups=1001(jru), 1100(teamtwo)
```

Jak widzimy, `jru` jest członkiem grup `jru` i `teamtwo`.

Więcej informacji o poleceniu `pw(8)` znaleźć można w jego podręczniku systemowym. Natomiast szczegółowe informacje o formacie pliku `/etc/group` znajdują się w podręczniku `group(5)`.

Przypisy

1. Chyba, że podepnijemy kilka terminali, co omawia Rozdział 22.
2. Możliwe jest również wykorzystanie identyfikatorów UID/GID do 4294967295, jednakże takie identyfikatory mogą być przyczyną poważnych problemów z oprogramowaniem, które dokonuje pewnych założeń co do wartości identyfikatorów.

Rozdział 14. Security

Much of this chapter has been taken from the security(7) manual page by Matthew Dillon.

14.1. Synopsis

This chapter will provide a basic introduction to system security concepts, some general good rules of thumb, and some advanced topics under FreeBSD. A lot of the topics covered here can be applied to system and Internet security in general as well. The Internet is no longer a „friendly” place in which everyone wants to be your kind neighbor. Securing your system is imperative to protect your data, intellectual property, time, and much more from the hands of hackers and the like.

FreeBSD provides an array of utilities and mechanisms to ensure the integrity and security of your system and network.

After reading this chapter, you will know:

- Basic system security concepts, in respect to FreeBSD.
- About the various crypt mechanisms available in FreeBSD, such as DES and MD5.
- How to set up one-time password authentication.
- How to configure TCP Wrappers for use with `inetd`.
- How to set up **KerberosIV** on FreeBSD releases prior to 5.0.
- How to set up **Kerberos5** on FreeBSD.
- How to configure IPsec and create a VPN between FreeBSD/Windows machines.
- How to configure and use **OpenSSH**, FreeBSD’s SSH implementation.
- What file system ACLs are and how to use them.
- How to use the **Portaudit** utility to audit third party software packages installed from the Ports Collection.
- How to utilize the FreeBSD security advisories publications.
- Have an idea of what Process Accounting is and how to enable it on FreeBSD.

Before reading this chapter, you should:

- Understand basic FreeBSD and Internet concepts.

Additional security topics are covered throughout this book. For example, Mandatory Access Control is discussed in Rozdział 15 and Internet Firewalls are discussed in Rozdział 26.

14.2. Introduction

Security is a function that begins and ends with the system administrator. While all BSD UNIX multi-user systems have some inherent security, the job of building and maintaining additional security mechanisms to keep those users „honest” is probably one of the single largest undertakings of the sysadmin. Machines are only as secure as you make them, and security concerns are ever competing with the human necessity for convenience. UNIX systems, in

general, are capable of running a huge number of simultaneous processes and many of these processes operate as servers — meaning that external entities can connect and talk to them. As yesterday’s mini-computers and mainframes become today’s desktops, and as computers become networked and internetwork, security becomes an even bigger issue.

Security is best implemented through a layered „onion” approach. In a nutshell, what you want to do is to create as many layers of security as are convenient and then carefully monitor the system for intrusions. You do not want to overbuild your security or you will interfere with the detection side, and detection is one of the single most important aspects of any security mechanism. For example, it makes little sense to set the `schg` flag (see `chflags(1)`) on every system binary because while this may temporarily protect the binaries, it prevents an attacker who has broken in from making an easily detectable change that may result in your security mechanisms not detecting the attacker at all.

System security also pertains to dealing with various forms of attack, including attacks that attempt to crash, or otherwise make a system unusable, but do not attempt to compromise the `root` account („break root”). Security concerns can be split up into several categories:

1. Denial of service attacks.
2. User account compromises.
3. Root compromise through accessible servers.
4. Root compromise via user accounts.
5. Backdoor creation.

A denial of service attack is an action that deprives the machine of needed resources. Typically, DoS attacks are brute-force mechanisms that attempt to crash or otherwise make a machine unusable by overwhelming its servers or network stack. Some DoS attacks try to take advantage of bugs in the networking stack to crash a machine with a single packet. The latter can only be fixed by applying a bug fix to the kernel. Attacks on servers can often be fixed by properly specifying options to limit the load the servers incur on the system under adverse conditions. Brute-force network attacks are harder to deal with. A spoofed-packet attack, for example, is nearly impossible to stop, short of cutting your system off from the Internet. It may not be able to take your machine down, but it can saturate your Internet connection.

A user account compromise is even more common than a DoS attack. Many sysadmins still run standard **telnetd**, **rlogind**, **rshd**, and **ftpd** servers on their machines. These servers, by default, do not operate over encrypted connections. The result is that if you have any moderate-sized user base, one or more of your users logging into your system from a remote location (which is the most common and convenient way to login to a system) will have his or her password sniffed. The attentive system admin will analyze his remote access logs looking for suspicious source addresses even for successful logins.

One must always assume that once an attacker has access to a user account, the attacker can break `root`. However, the reality is that in a well secured and maintained system, access to a user account does not necessarily give the attacker access to `root`. The distinction is important because without access to `root` the attacker cannot generally hide his tracks and may, at best, be able to do nothing more than mess with the user’s files, or crash the machine. User account compromises are very common because users tend not to take the precautions that sysadmins take.

System administrators must keep in mind that there are potentially many ways to break `root` on a machine. The attacker may know the `root` password, the attacker may find a bug in a root-run server and be able to break `root` over a network connection to that server, or the attacker may know of a bug in a `suid-root` program that allows the attacker to break `root` once he has broken into a user’s account. If an attacker has found a way to break `root` on a machine, the attacker may not have a need to install a backdoor. Many of the `root` holes found and closed to date involve a considerable amount of work by the attacker to cleanup after himself, so most attackers install backdoors.

A backdoor provides the attacker with a way to easily regain `root` access to the system, but it also gives the smart system administrator a convenient way to detect the intrusion. Making it impossible for an attacker to install a backdoor may actually be detrimental to your security, because it will not close off the hole the attacker found to break in the first place.

Security remedies should always be implemented with a multi-layered „onion peel” approach and can be categorized as follows:

1. Securing `root` and staff accounts.
2. Securing `root`-run servers and `suid/sgid` binaries.
3. Securing user accounts.
4. Securing the password file.
5. Securing the kernel core, raw devices, and file systems.
6. Quick detection of inappropriate changes made to the system.
7. Paranoia.

The next section of this chapter will cover the above bullet items in greater depth.

14.3. Securing FreeBSD

Command vs. Protocol: Throughout this document, we will use **bold** text to refer to an application, and a `monospaced` font to refer to specific commands. Protocols will use a normal font. This typographical distinction is useful for instances such as `ssh`, since it is a protocol as well as command.

The sections that follow will cover the methods of securing your FreeBSD system that were mentioned in the last section of this chapter.

14.3.1. Securing the `root` Account and Staff Accounts

First off, do not bother securing staff accounts if you have not secured the `root` account. Most systems have a password assigned to the `root` account. The first thing you do is assume that the password is *always* compromised. This does not mean that you should remove the password. The password is almost always necessary for console access to the machine. What it does mean is that you should not make it possible to use the password outside of the console or possibly even with the `su(1)` command. For example, make sure that your `ptys` are specified as being insecure in the `/etc/ttys` file so that direct `root` logins via `telnet` or `rlogin` are disallowed. If using other login services such as **sshd**, make sure that direct `root` logins are disabled there as well. You can do this by editing your `/etc/ssh/sshd_config` file, and making sure that `PermitRootLogin` is set to `NO`. Consider every access method — services such as FTP often fall through the cracks. Direct `root` logins should only be allowed via the system console.

Of course, as a sysadmin you have to be able to get to `root`, so we open up a few holes. But we make sure these holes require additional password verification to operate. One way to make `root` accessible is to add appropriate staff accounts to the `wheel` group (in `/etc/group`). The staff members placed in the `wheel` group are allowed to `su` to `root`. You should never give staff members native `wheel` access by putting them in the `wheel` group in their

password entry. Staff accounts should be placed in a `staff` group, and then added to the `wheel` group via the `/etc/group` file. Only those staff members who actually need to have `root` access should be placed in the `wheel` group. It is also possible, when using an authentication method such as Kerberos, to use Kerberos' `.k5login` file in the `root` account to allow a `ksu(1)` to `root` without having to place anyone at all in the `wheel` group. This may be the better solution since the `wheel` mechanism still allows an intruder to break `root` if the intruder has gotten hold of your password file and can break into a staff account. While having the `wheel` mechanism is better than having nothing at all, it is not necessarily the safest option.

An indirect way to secure staff accounts, and ultimately `root` access is to use an alternative login access method and do what is known as „starring” out the encrypted password for the staff accounts. Using the `vipw(8)` command, one can replace each instance of an encrypted password with a single „*” character. This command will update the `/etc/master.passwd` file and user/password database to disable password-authenticated logins.

A staff account entry such as:

```
foobar:R9DT/Fa1/LV9U:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

Should be changed to this:

```
foobar:*:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

This change will prevent normal logins from occurring, since the encrypted password will never match „*”. With this done, staff members must use another mechanism to authenticate themselves such as `kerberos(1)` or `ssh(1)` using a public/private key pair. When using something like Kerberos, one generally must secure the machines which run the Kerberos servers and your desktop workstation. When using a public/private key pair with `ssh`, one must generally secure the machine used to login *from* (typically one's workstation). An additional layer of protection can be added to the key pair by password protecting the key pair when creating it with `ssh-keygen(1)`. Being able to „star” out the passwords for staff accounts also guarantees that staff members can only login through secure access methods that you have set up. This forces all staff members to use secure, encrypted connections for all of their sessions, which closes an important hole used by many intruders: sniffing the network from an unrelated, less secure machine.

The more indirect security mechanisms also assume that you are logging in from a more restrictive server to a less restrictive server. For example, if your main box is running all sorts of servers, your workstation should not be running any. In order for your workstation to be reasonably secure you should run as few servers as possible, up to and including no servers at all, and you should run a password-protected screen blanker. Of course, given physical access to a workstation an attacker can break any sort of security you put on it. This is definitely a problem that you should consider, but you should also consider the fact that the vast majority of break-ins occur remotely, over a network, from people who do not have physical access to your workstation or servers.

Using something like Kerberos also gives you the ability to disable or change the password for a staff account in one place, and have it immediately affect all the machines on which the staff member may have an account. If a staff member's account gets compromised, the ability to instantly change his password on all machines should not be underrated. With discrete passwords, changing a password on *N* machines can be a mess. You can also impose re-passwording restrictions with Kerberos: not only can a Kerberos ticket be made to timeout after a while, but the Kerberos system can require that the user choose a new password after a certain period of time (say, once a month).

14.3.2. Securing Root-run Servers and SUID/SGID Binaries

The prudent sysadmin only runs the servers he needs to, no more, no less. Be aware that third party servers are often the most bug-prone. For example, running an old version of **imapd** or **popper** is like giving a universal `root` ticket out to the entire world. Never run a server that you have not checked out carefully. Many servers do not need to be

run as `root`. For example, the **ntalk**, **comsat**, and **finger** daemons can be run in special user *sandboxes*. A sandbox is not perfect, unless you go through a large amount of trouble, but the onion approach to security still stands: If someone is able to break in through a server running in a sandbox, they still have to break out of the sandbox. The more layers the attacker must break through, the lower the likelihood of his success. Root holes have historically been found in virtually every server ever run as `root`, including basic system servers. If you are running a machine through which people only login via **sshd** and never login via **telnetd** or **rshd** or **rlogind**, then turn off those services!

FreeBSD now defaults to running **ntalkd**, **comsat**, and **finger** in a sandbox. Another program which may be a candidate for running in a sandbox is **named(8)**. `/etc/defaults/rc.conf` includes the arguments necessary to run **named** in a sandbox in a commented-out form. Depending on whether you are installing a new system or upgrading an existing system, the special user accounts used by these sandboxes may not be installed. The prudent sysadmin would research and implement sandboxes for servers whenever possible.

There are a number of other servers that typically do not run in sandboxes: **sendmail**, **popper**, **imapd**, **ftpd**, and others. There are alternatives to some of these, but installing them may require more work than you are willing to perform (the convenience factor strikes again). You may have to run these servers as `root` and rely on other mechanisms to detect break-ins that might occur through them.

The other big potential `root` holes in a system are the `suid-root` and `sgid` binaries installed on the system. Most of these binaries, such as **rlogin**, reside in `/bin`, `/sbin`, `/usr/bin`, or `/usr/sbin`. While nothing is 100% safe, the system-default `suid` and `sgid` binaries can be considered reasonably safe. Still, `root` holes are occasionally found in these binaries. A `root` hole was found in `xlib` in 1998 that made **xterm** (which is typically `suid`) vulnerable. It is better to be safe than sorry and the prudent sysadmin will restrict `suid` binaries, that only staff should run, to a special group that only staff can access, and get rid of (`chmod 000`) any `suid` binaries that nobody uses. A server with no display generally does not need an **xterm** binary. `Sgid` binaries can be almost as dangerous. If an intruder can break an `sgid-kmem` binary, the intruder might be able to read `/dev/kmem` and thus read the encrypted password file, potentially compromising any passworded account. Alternatively an intruder who breaks group `kmem` can monitor keystrokes sent through `ptys`, including `ptys` used by users who login through secure methods. An intruder that breaks the `tty` group can write to almost any user's `tty`. If a user is running a terminal program or emulator with a keyboard-simulation feature, the intruder can potentially generate a data stream that causes the user's terminal to echo a command, which is then run as that user.

14.3.3. Securing User Accounts

User accounts are usually the most difficult to secure. While you can impose Draconian access restrictions on your staff and „star” out their passwords, you may not be able to do so with any general user accounts you might have. If you do have sufficient control, then you may win out and be able to secure the user accounts properly. If not, you simply have to be more vigilant in your monitoring of those accounts. Use of `ssh` and Kerberos for user accounts is more problematic, due to the extra administration and technical support required, but still a very good solution compared to a crypted password file.

14.3.4. Securing the Password File

The only sure fire way is to * out as many passwords as you can and use `ssh` or Kerberos for access to those accounts. Even though the encrypted password file (`/etc/spwd.db`) can only be read by `root`, it may be possible for an intruder to obtain read access to that file even if the attacker cannot obtain `root-write` access.

Your security scripts should always check for and report changes to the password file (see the Checking file integrity section below).

14.3.5. Securing the Kernel Core, Raw Devices, and File systems

If an attacker breaks `root` he can do just about anything, but there are certain conveniences. For example, most modern kernels have a packet sniffing device driver built in. Under FreeBSD it is called the `bpf` device. An intruder will commonly attempt to run a packet sniffer on a compromised machine. You do not need to give the intruder the capability and most systems do not have the need for the `bpf` device compiled in.

But even if you turn off the `bpf` device, you still have `/dev/mem` and `/dev/kmem` to worry about. For that matter, the intruder can still write to raw disk devices. Also, there is another kernel feature called the module loader, `kldload(8)`. An enterprising intruder can use a KLD module to install his own `bpf` device, or other sniffing device, on a running kernel. To avoid these problems you have to run the kernel at a higher secure level, at least `securelevel 1`. The `securelevel` can be set with a `sysctl` on the `kern.securelevel` variable. Once you have set the `securelevel` to 1, write access to raw devices will be denied and special `chflags` flags, such as `schg`, will be enforced. You must also ensure that the `schg` flag is set on critical startup binaries, directories, and script files — everything that gets run up to the point where the `securelevel` is set. This might be overdoing it, and upgrading the system is much more difficult when you operate at a higher secure level. You may compromise and run the system at a higher secure level but not set the `schg` flag for every system file and directory under the sun. Another possibility is to simply mount `/` and `/usr` read-only. It should be noted that being too Draconian in what you attempt to protect may prevent the all-important detection of an intrusion.

14.3.6. Checking File Integrity: Binaries, Configuration Files, Etc.

When it comes right down to it, you can only protect your core system configuration and control files so much before the convenience factor rears its ugly head. For example, using `chflags` to set the `schg` bit on most of the files in `/` and `/usr` is probably counterproductive, because while it may protect the files, it also closes a detection window. The last layer of your security onion is perhaps the most important — detection. The rest of your security is pretty much useless (or, worse, presents you with a false sense of safety) if you cannot detect potential incursions. Half the job of the onion is to slow down the attacker, rather than stop him, in order to give the detection side of the equation a chance to catch him in the act.

The best way to detect an incursion is to look for modified, missing, or unexpected files. The best way to look for modified files is from another (often centralized) limited-access system. Writing your security scripts on the extra-secure limited-access system makes them mostly invisible to potential attackers, and this is important. In order to take maximum advantage you generally have to give the limited-access box significant access to the other machines in the business, usually either by doing a read-only NFS export of the other machines to the limited-access box, or by setting up `ssh` key-pairs to allow the limited-access box to `ssh` to the other machines. Except for its network traffic, NFS is the least visible method — allowing you to monitor the file systems on each client box virtually undetected. If your limited-access server is connected to the client boxes through a switch, the NFS method is often the better choice. If your limited-access server is connected to the client boxes through a hub, or through several layers of routing, the NFS method may be too insecure (network-wise) and using `ssh` may be the better choice even with the audit-trail tracks that `ssh` lays.

Once you give a limited-access box, at least read access to the client systems it is supposed to monitor, you must write scripts to do the actual monitoring. Given an NFS mount, you can write scripts out of simple system utilities such as `find(1)` and `md5(1)`. It is best to physically `md5` the client-box files at least once a day, and to test control files such as those found in `/etc` and `/usr/local/etc` even more often. When mismatches are found, relative to the base `md5` information the limited-access machine knows is valid, it should scream at a `sysadmin` to go check it out. A good security script will also check for inappropriate `suid` binaries and for new or deleted files on system partitions such as `/` and `/usr`.

When using `ssh` rather than NFS, writing the security script is much more difficult. You essentially have to `scp` the scripts to the client box in order to run them, making them visible, and for safety you also need to `scp` the binaries (such as `find`) that those scripts use. The `ssh` client on the client box may already be compromised. All in all, using `ssh` may be necessary when running over insecure links, but it is also a lot harder to deal with.

A good security script will also check for changes to user and staff members access configuration files: `.rhosts`, `.shosts`, `.ssh/authorized_keys` and so forth... files that might fall outside the purview of the MD5 check.

If you have a huge amount of user disk space, it may take too long to run through every file on those partitions. In this case, setting mount flags to disallow `suid` binaries and devices on those partitions is a good idea. The `nodev` and `nosuid` options (see `mount(8)`) are what you want to look into. You should probably scan them anyway, at least once a week, since the object of this layer is to detect a break-in whether or not the break-in is effective.

Process accounting (see `accton(8)`) is a relatively low-overhead feature of the operating system which might help as a post-break-in evaluation mechanism. It is especially useful in tracking down how an intruder has actually broken into a system, assuming the file is still intact after the break-in occurs.

Finally, security scripts should process the log files, and the logs themselves should be generated in as secure a manner as possible — remote `syslog` can be very useful. An intruder tries to cover his tracks, and log files are critical to the `sysadmin` trying to track down the time and method of the initial break-in. One way to keep a permanent record of the log files is to run the system console to a serial port and collect the information on a continuing basis through a secure machine monitoring the consoles.

14.3.7. Paranoia

A little paranoia never hurts. As a rule, a `sysadmin` can add any number of security features, as long as they do not affect convenience, and can add security features that *do* affect convenience with some added thought. Even more importantly, a security administrator should mix it up a bit — if you use recommendations such as those given by this document verbatim, you give away your methodologies to the prospective attacker who also has access to this document.

14.3.8. Denial of Service Attacks

This section covers Denial of Service attacks. A DoS attack is typically a packet attack. While there is not much you can do about modern spoofed packet attacks that saturate your network, you can generally limit the damage by ensuring that the attacks cannot take down your servers.

1. Limiting server forks.
2. Limiting springboard attacks (ICMP response attacks, ping broadcast, etc.).
3. Kernel Route Cache.

A common DoS attack is against a forking server that attempts to cause the server to eat processes, file descriptors, and memory, until the machine dies. **inetd** (see `inetd(8)`) has several options to limit this sort of attack. It should be noted that while it is possible to prevent a machine from going down, it is not generally possible to prevent a service from being disrupted by the attack. Read the **inetd** manual page carefully and pay specific attention to the `-c`, `-C`, and `-R` options. Note that spoofed-IP attacks will circumvent the `-C` option to **inetd**, so typically a combination of options must be used. Some standalone servers have self-fork-limitation parameters.

Sendmail has its `-OMaxDaemonChildren` option, which tends to work much better than trying to use sendmail's load limiting options due to the load lag. You should specify a `MaxDaemonChildren` parameter, when you start **sendmail**, high enough to handle your expected load, but not so high that the computer cannot handle that number of **sendmails** without falling on its face. It is also prudent to run sendmail in queued mode (`-ODeliveryMode=queued`) and to run the daemon (`sendmail -bd`) separate from the queue-runs (`sendmail -q15m`). If you still want real-time delivery you can run the queue at a much lower interval, such as `-q1m`, but be sure to specify a reasonable `MaxDaemonChildren` option for *that* sendmail to prevent cascade failures.

Syslogd can be attacked directly and it is strongly recommended that you use the `-s` option whenever possible, and the `-a` option otherwise.

You should also be fairly careful with connect-back services such as **TCP Wrapper**'s reverse-identd, which can be attacked directly. You generally do not want to use the reverse-ident feature of **TCP Wrapper** for this reason.

It is a very good idea to protect internal services from external access by firewalling them off at your border routers. The idea here is to prevent saturation attacks from outside your LAN, not so much to protect internal services from network-based `root` compromise. Always configure an exclusive firewall, i.e., „firewall everything *except* ports A, B, C, D, and M-Z". This way you can firewall off all of your low ports except for certain specific services such as **named** (if you are primary for a zone), **ntalkd**, **sendmail**, and other Internet-accessible services. If you try to configure the firewall the other way — as an inclusive or permissive firewall, there is a good chance that you will forget to „close" a couple of services, or that you will add a new internal service and forget to update the firewall. You can still open up the high-numbered port range on the firewall, to allow permissive-like operation, without compromising your low ports. Also take note that FreeBSD allows you to control the range of port numbers used for dynamic binding, via the various `net.inet.ip.portrange` `sysctl`'s (`sysctl -a | fgrep portrange`), which can also ease the complexity of your firewall's configuration. For example, you might use a normal first/last range of 4000 to 5000, and a `hiport` range of 49152 to 65535, then block off everything under 4000 in your firewall (except for certain specific Internet-accessible ports, of course).

Another common DoS attack is called a springboard attack — to attack a server in a manner that causes the server to generate responses which overloads the server, the local network, or some other machine. The most common attack of this nature is the *ICMP ping broadcast attack*. The attacker spoofs ping packets sent to your LAN's broadcast address with the source IP address set to the actual machine they wish to attack. If your border routers are not configured to stomp on ping's to broadcast addresses, your LAN winds up generating sufficient responses to the spoofed source address to saturate the victim, especially when the attacker uses the same trick on several dozen broadcast addresses over several dozen different networks at once. Broadcast attacks of over a hundred and twenty megabits have been measured. A second common springboard attack is against the ICMP error reporting system. By constructing packets that generate ICMP error responses, an attacker can saturate a server's incoming network and cause the server to saturate its outgoing network with ICMP responses. This type of attack can also crash the server by running it out of `mbuf`'s, especially if the server cannot drain the ICMP responses it generates fast enough. Use the `sysctl` variable `net.inet.icmp.icmplim` to limit these attacks. The last major class of springboard attacks is related to certain internal **inetd** services such as the `udp echo` service. An attacker simply spoofs a UDP packet with the source address being server A's echo port, and the destination address being server B's echo port, where server A and B are both on your LAN. The two servers then bounce this one packet back and forth between each other. The attacker can overload both servers and their LANs simply by injecting a few packets in this manner. Similar problems exist with the internal **chargen** port. A competent sysadmin will turn off all of these `inetd`-internal test services.

Spoofed packet attacks may also be used to overload the kernel route cache. Refer to the `net.inet.ip.rtxpire`, `rtminexpire`, and `rtmaxcache` `sysctl` parameters. A spoofed packet attack that uses a random source IP will cause the kernel to generate a temporary cached route in the route table, viewable with `netstat -rna | fgrep w3`. These routes typically timeout in 1600 seconds or so. If the kernel detects that the cached route table has gotten too big it will dynamically reduce the `rtxpire` but will never decrease it to less than `rtminexpire`. There are two

problems:

1. The kernel does not react quickly enough when a lightly loaded server is suddenly attacked.
2. The `rtminexpire` is not low enough for the kernel to survive a sustained attack.

If your servers are connected to the Internet via a T3 or better, it may be prudent to manually override both `rtexpire` and `rtminexpire` via `sysctl(8)`. Never set either parameter to zero (unless you want to crash the machine). Setting both parameters to 2 seconds should be sufficient to protect the route table from attack.

14.3.9. Access Issues with Kerberos and SSH

There are a few issues with both Kerberos and `ssh` that need to be addressed if you intend to use them. Kerberos V is an excellent authentication protocol, but there are bugs in the kerberized **telnet** and **rlogin** applications that make them unsuitable for dealing with binary streams. Also, by default Kerberos does not encrypt a session unless you use the `-x` option. **ssh** encrypts everything by default.

`ssh` works quite well in every respect except that it forwards encryption keys by default. What this means is that if you have a secure workstation holding keys that give you access to the rest of the system, and you `ssh` to an insecure machine, your keys are usable. The actual keys themselves are not exposed, but `ssh` installs a forwarding port for the duration of your login, and if an attacker has broken `root` on the insecure machine he can utilize that port to use your keys to gain access to any other machine that your keys unlock.

We recommend that you use `ssh` in combination with Kerberos whenever possible for staff logins. **ssh** can be compiled with Kerberos support. This reduces your reliance on potentially exposed `ssh` keys while at the same time protecting passwords via Kerberos. `ssh` keys should only be used for automated tasks from secure machines (something that Kerberos is unsuited to do). We also recommend that you either turn off key-forwarding in the `ssh` configuration, or that you make use of the `from=IP/DOMAIN` option that `ssh` allows in its `authorized_keys` file to make the key only usable to entities logging in from specific machines.

14.4. DES, MD5, and Crypt

Parts rewritten and updated by Bill Swingle.

Every user on a UNIX system has a password associated with their account. It seems obvious that these passwords need to be known only to the user and the actual operating system. In order to keep these passwords secret, they are encrypted with what is known as a „one-way hash”, that is, they can only be easily encrypted but not decrypted. In other words, what we told you a moment ago was obvious is not even true: the operating system itself does not *really* know the password. It only knows the *encrypted* form of the password. The only way to get the „plain-text” password is by a brute force search of the space of possible passwords.

Unfortunately the only secure way to encrypt passwords when UNIX came into being was based on DES, the Data Encryption Standard. This was not such a problem for users resident in the US, but since the source code for DES could not be exported outside the US, FreeBSD had to find a way to both comply with US law and retain compatibility with all the other UNIX variants that still used DES.

The solution was to divide up the encryption libraries so that US users could install the DES libraries and use DES but international users still had an encryption method that could be exported abroad. This is how FreeBSD came to

use MD5 as its default encryption method. MD5 is believed to be more secure than DES, so installing DES is offered primarily for compatibility reasons.

14.4.1. Recognizing Your Crypt Mechanism

Currently the library supports DES, MD5 and Blowfish hash functions. By default FreeBSD uses MD5 to encrypt passwords.

It is pretty easy to identify which encryption method FreeBSD is set up to use. Examining the encrypted passwords in the `/etc/master.passwd` file is one way. Passwords encrypted with the MD5 hash are longer than those encrypted with the DES hash and also begin with the characters `1`. Passwords starting with `$2a$` are encrypted with the Blowfish hash function. DES password strings do not have any particular identifying characteristics, but they are shorter than MD5 passwords, and are coded in a 64-character alphabet which does not include the `$` character, so a relatively short string which does not begin with a dollar sign is very likely a DES password.

The password format used for new passwords is controlled by the `passwd_format` login capability in `/etc/login.conf`, which takes values of `des`, `md5` or `blf`. See the `login.conf(5)` manual page for more information about login capabilities.

14.5. One-time Passwords

By default, FreeBSD includes support for OPIE (One-time Passwords In Everything), which uses the MD5 hash by default.

There are three different sorts of passwords which we will discuss below. The first is your usual UNIX style or Kerberos password; we will call this a „UNIX password”. The second sort is the one-time password which is generated by the OPIE `opiekey(1)` program and accepted by the `opiepasswd(1)` program and the login prompt; we will call this a „one-time password”. The final sort of password is the secret password which you give to the `opiekey` program (and sometimes the `opiepasswd` programs) which it uses to generate one-time passwords; we will call it a „secret password” or just unqualified „password”.

The secret password does not have anything to do with your UNIX password; they can be the same but this is not recommended. OPIE secret passwords are not limited to 8 characters like old UNIX passwords¹, they can be as long as you like. Passwords of six or seven word long phrases are fairly common. For the most part, the OPIE system operates completely independently of the UNIX password system.

Besides the password, there are two other pieces of data that are important to OPIE. One is what is known as the „seed” or „key”, consisting of two letters and five digits. The other is what is called the „iteration count”, a number between 1 and 100. OPIE creates the one-time password by concatenating the seed and the secret password, then applying the MD5 hash as many times as specified by the iteration count and turning the result into six short English words. These six English words are your one-time password. The authentication system (primarily PAM) keeps track of the last one-time password used, and the user is authenticated if the hash of the user-provided password is equal to the previous password. Because a one-way hash is used it is impossible to generate future one-time passwords if a successfully used password is captured; the iteration count is decremented after each successful login to keep the user and the login program in sync. When the iteration count gets down to 1, OPIE must be reinitialized.

There are a few programs involved in each system which we will discuss below. The `opiekey` program accepts an iteration count, a seed, and a secret password, and generates a one-time password or a consecutive list of one-time passwords. The `opiepasswd` program is used to initialize OPIE, and to change passwords, iteration counts, or seeds; it takes either a secret passphrase, or an iteration count, seed, and a one-time password. The `opieinfo` program will

examine the relevant credentials files (`/etc/otpkeys`) and print out the invoking user's current iteration count and seed.

There are four different sorts of operations we will cover. The first is using `otppasswd` over a secure connection to set up one-time-passwords for the first time, or to change your password or seed. The second operation is using `otppasswd` over an insecure connection, in conjunction with `otpkey` over a secure connection, to do the same. The third is using `otpkey` to log in over an insecure connection. The fourth is using `otpkey` to generate a number of keys which can be written down or printed out to carry with you when going to some location without secure connections to anywhere.

14.5.1. Secure Connection Initialization

To initialize OPIE for the first time, execute the `otppasswd` command:

```
% otplib -c
[grimreaper] ~ $ otplib -f -c
Adding unfurl:
Only use this method from the console; NEVER from remote. If you are using
telnet, xterm, or a dial-in, type ^C now or exit with no password.
Then run otplib without the -c parameter.
Using MD5 to compute responses.
Enter new secret pass phrase:
Again new secret pass phrase:
ID unfurl OTP key is 499 to4268
MOS MALL GOAT ARM AVID COED
```

At the `Enter new secret pass phrase:` or `Enter secret password:` prompts, you should enter a password or phrase. Remember, this is not the password that you will use to login with, this is used to generate your one-time login keys. The „ID” line gives the parameters of your particular instance: your login name, the iteration count, and seed. When logging in the system will remember these parameters and present them back to you so you do not have to remember them. The last line gives the particular one-time password which corresponds to those parameters and your secret password; if you were to re-login immediately, this one-time password is the one you would use.

14.5.2. Insecure Connection Initialization

To initialize or change your secret password over an insecure connection, you will need to already have a secure connection to some place where you can run `otpkey`; this might be in the form of a shell prompt on a machine you trust. You will also need to make up an iteration count (100 is probably a good value), and you may make up your own seed or use a randomly-generated one. Over on the insecure connection (to the machine you are initializing), use `otplib`:

```
% otplib

Updating unfurl:
You need the response from an OTP generator.
Old secret pass phrase:
    otp-md5 498 to4268 ext
    Response: GAME GAG WELT OUT DOWN CHAT
New secret pass phrase:
    otp-md5 499 to4269
```

```
Response: LINE PAP MILK NELL BUOY TROY
```

```
ID mark OTP key is 499 gr4269
LINE PAP MILK NELL BUOY TROY
```

To accept the default seed press **Return**. Then before entering an access password, move over to your secure connection and give it the same parameters:

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

Now switch back over to the insecure connection, and copy the one-time password generated over to the relevant program.

14.5.3. Generating a Single One-time Password

Once you have initialized OPIE and login, you will be presented with a prompt like this:

```
% telnet example.com
Trying 10.0.0.1...
Connected to example.com
Escape character is '^]'.

FreeBSD/i386 (example.com) (tttya)

login: <username>
otp-md5 498 gr4269 ext
Password:
```

As a side note, the OPIE prompts have a useful feature (not shown here): if you press **Return** at the password prompt, the prompter will turn echo on, so you can see what you are typing. This can be extremely useful if you are attempting to type in a password by hand, such as from a printout.

At this point you need to generate your one-time password to answer this login prompt. This must be done on a trusted system that you can run `opiekey` on. (There are versions of these for DOS, Windows and Mac OS as well.) They need the iteration count and the seed as command line options. You can cut-and-paste these right from the login prompt on the machine that you are logging in to.

On the trusted system:

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

Now that you have your one-time password you can continue logging in.

14.5.4. Generating Multiple One-time Passwords

Sometimes you have to go places where you do not have access to a trusted machine or secure connection. In this case, it is possible to use the `opiekey` command to generate a number of one-time passwords beforehand to be printed out and taken with you. For example:

```
% opiekey -n 5 30 zz99999
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase: <secret password>
26: JOAN BORE FOSS DES NAY QUIT
27: LATE BIAS SLAY FOLK MUCH TRIG
28: SALT TIN ANTI LOON NEAL USE
29: RIO ODIN GO BYE FURY TIC
30: GREW JIVE SAN GIRD BOIL PHI
```

The `-n 5` requests five keys in sequence, the `30` specifies what the last iteration number should be. Note that these are printed out in *reverse* order of eventual use. If you are really paranoid, you might want to write the results down by hand; otherwise you can cut-and-paste into `lpr`. Note that each line shows both the iteration count and the one-time password; you may still find it handy to scratch off passwords as you use them.

14.5.5. Restricting Use of UNIX® Passwords

OPIE can restrict the use of UNIX passwords based on the IP address of a login session. The relevant file is `/etc/opieaccess`, which is present by default. Please check `opieaccess(5)` for more information on this file and which security considerations you should be aware of when using it.

Here is a sample `opieaccess` file:

```
permit 192.168.0.0 255.255.0.0
```

This line allows users whose IP source address (which is vulnerable to spoofing) matches the specified value and mask, to use UNIX passwords at any time.

If no rules in `opieaccess` are matched, the default is to deny non-OPIE logins.

14.6. TCP Wrappers

Written by: Tom Rhodes.

Anyone familiar with `inetd(8)` has probably heard of TCP Wrappers at some point. But few individuals seem to fully comprehend its usefulness in a network environment. It seems that everyone wants to install a firewall to handle network connections. While a firewall has a wide variety of uses, there are some things that a firewall not handle such as sending text back to the connection originator. The TCP software does this and much more. In the next few sections many of the TCP Wrappers features will be discussed, and, when applicable, example configuration lines will be provided.

The TCP Wrappers software extends the abilities of `inetd` to provide support for every server daemon under its control. Using this method it is possible to provide logging support, return messages to connections, permit a daemon

to only accept internal connections, etc. While some of these features can be provided by implementing a firewall, this will add not only an extra layer of protection but go beyond the amount of control a firewall can provide.

The added functionality of TCP Wrappers should not be considered a replacement for a good firewall. TCP Wrappers can be used in conjunction with a firewall or other security enhancements though and it can serve nicely as an extra layer of protection for the system.

Since this is an extension to the configuration of `inetd`, the reader is expected have read the `inetd` configuration section.

Notatka: While programs run by `inetd(8)` are not exactly „daemons”, they have traditionally been called daemons. This is the term we will use in this section too.

14.6.1. Initial Configuration

The only requirement of using TCP Wrappers in FreeBSD is to ensure the `inetd` server is started from `rc.conf` with the `-ww` option; this is the default setting. Of course, proper configuration of `/etc/hosts.allow` is also expected, but `syslogd(8)` will throw messages in the system logs in these cases.

Notatka: Unlike other implementations of TCP Wrappers, the use of `hosts.deny` has been deprecated. All configuration options should be placed in `/etc/hosts.allow`.

In the simplest configuration, daemon connection policies are set to either be permitted or blocked depending on the options in `/etc/hosts.allow`. The default configuration in FreeBSD is to allow a connection to every daemon started with `inetd`. Changing this will be discussed only after the basic configuration is covered.

Basic configuration usually takes the form of `daemon : address : action`. Where `daemon` is the daemon name which `inetd` started. The `address` can be a valid hostname, an IP address or an IPv6 address enclosed in brackets (`[]`). The `action` field can be either `allow` or `deny` to grant or deny access appropriately. Keep in mind that configuration works off a first rule match semantic, meaning that the configuration file is scanned in ascending order for a matching rule. When a match is found the rule is applied and the search process will halt.

Several other options exist but they will be explained in a later section. A simple configuration line may easily be constructed from that information alone. For example, to allow POP3 connections via the `mail/qpopper` daemon, the following lines should be appended to `hosts.allow`:

```
# This line is required for POP3 connections:
qpopper : ALL : allow
```

After adding this line, `inetd` will need restarted. This can be accomplished by use of the `kill(1)` command, or with the `restart` parameter with `/etc/rc.d/inetd`.

14.6.2. Advanced Configuration

TCP Wrappers has advanced options too; they will allow for more control over the way connections are handled. In some cases it may be a good idea to return a comment to certain hosts or daemon connections. In other cases, perhaps a log file should be recorded or an email sent to the administrator. Other situations may require the use of a service

for local connections only. This is all possible through the use of configuration options known as wildcards, expansion characters and external command execution. The next two sections are written to cover these situations.

14.6.2.1. External Commands

Suppose that a situation occurs where a connection should be denied yet a reason should be sent to the individual who attempted to establish that connection. How could it be done? That action can be made possible by using the `twist` option. When a connection attempt is made, `twist` will be called to execute a shell command or script. An example already exists in the `hosts.allow` file:

```
# The rest of the daemons are protected.
ALL : ALL \
    : severity auth.info \
    : twist /bin/echo "You are not welcome to use %d from %h."
```

This example shows that the message, „You are not allowed to use daemon from hostname.” will be returned for any daemon not previously configured in the access file. This is extremely useful for sending a reply back to the connection initiator right after the established connection is dropped. Note that any message returned *must* be wrapped in quote " characters; there are no exceptions to this rule.

Ostrzeżenie! It may be possible to launch a denial of service attack on the server if an attacker, or group of attackers could flood these daemons with connection requests.

Another possibility is to use the `spawn` option in these cases. Like `twist`, the `spawn` implicitly denies the connection and may be used to run external shell commands or scripts. Unlike `twist`, `spawn` will not send a reply back to the individual who established the connection. For an example, consider the following configuration line:

```
# We do not allow connections from example.com:
ALL : .example.com \
    : spawn (/bin/echo %a from %h attempted to access %d >> \
    /var/log/connections.log) \
    : deny
```

This will deny all connection attempts from the `*.example.com` domain; simultaneously logging the hostname, IP address and the daemon which they attempted to access in the `/var/log/connections.log` file.

Aside from the already explained substitution characters above, e.g. `%a`, a few others exist. See the `hosts_access(5)` manual page for the complete list.

14.6.2.2. Wildcard Options

Thus far the `ALL` example has been used continuously throughout the examples. Other options exist which could extend the functionality a bit further. For instance, `ALL` may be used to match every instance of either a daemon, domain or an IP address. Another wildcard available is `PARANOID` which may be used to match any host which provides an IP address that may be forged. In other words, `paranoid` may be used to define an action to be taken whenever a connection is made from an IP address that differs from its hostname. The following example may shed some more light on this discussion:

```
# Block possibly spoofed requests to sendmail:
```

```
sendmail : PARANOID : deny
```

In that example all connection requests to `sendmail` which have an IP address that varies from its hostname will be denied.

Uwaga! Using the `PARANOID` may severely cripple servers if the client or server has a broken DNS setup. Administrator discretion is advised.

To learn more about wildcards and their associated functionality, see the `hosts_access(5)` manual page.

Before any of the specific configuration lines above will work, the first configuration line should be commented out in `hosts.allow`. This was noted at the beginning of this section.

14.7. KerberosIV

Contributed by Mark Murray. Based on a contribution by Mark Dapoz.

Kerberos is a network add-on system/protocol that allows users to authenticate themselves through the services of a secure server. Services such as remote login, remote copy, secure inter-system file copying and other high-risk tasks are made considerably safer and more controllable.

The following instructions can be used as a guide on how to set up Kerberos as distributed for FreeBSD. However, you should refer to the relevant manual pages for a complete description.

14.7.1. Installing KerberosIV

Kerberos is an optional component of FreeBSD. The easiest way to install this software is by selecting the `krb4` or `krb5` distribution in **`sysinstall`** during the initial installation of FreeBSD. This will install the „eBones” (KerberosIV) or „Heimdal” (Kerberos5) implementation of Kerberos. These implementations are included because they are developed outside the USA/Canada and were thus available to system owners outside those countries during the era of restrictive export controls on cryptographic code from the USA.

Alternatively, the MIT implementation of Kerberos is available from the Ports Collection as `security/krb5`.

14.7.2. Creating the Initial Database

This is done on the Kerberos server only. First make sure that you do not have any old Kerberos databases around. You should change to the directory `/etc/kerberosIV` and check that only the following files are present:

```
# cd /etc/kerberosIV
# ls
README          krb.conf        krb.realms
```

If any additional files (such as `principal.*` or `master_key`) exist, then use the `kdb_destroy` command to destroy the old Kerberos database, or if Kerberos is not running, simply delete the extra files.

You should now edit the `krb.conf` and `krb.realms` files to define your Kerberos realm. In this case the realm will be `EXAMPLE.COM` and the server is `grunt.example.com`. We edit or create the `krb.conf` file:

```
# cat krb.conf
EXAMPLE.COM
EXAMPLE.COM grunt.example.com admin server
CS.BERKELEY.EDU okeeffe.berkeley.edu
ATHENA.MIT.EDU kerberos.mit.edu
ATHENA.MIT.EDU kerberos-1.mit.edu
ATHENA.MIT.EDU kerberos-2.mit.edu
ATHENA.MIT.EDU kerberos-3.mit.edu
LCS.MIT.EDU kerberos.lcs.mit.edu
TELECOM.MIT.EDU bitsy.mit.edu
ARC.NASA.GOV trident.arc.nasa.gov
```

In this case, the other realms do not need to be there. They are here as an example of how a machine may be made aware of multiple realms. You may wish to not include them for simplicity.

The first line names the realm in which this system works. The other lines contain realm/host entries. The first item on a line is a realm, and the second is a host in that realm that is acting as a „key distribution center”. The words `admin server` following a host’s name means that host also provides an administrative database server. For further explanation of these terms, please consult the Kerberos manual pages.

Now we have to add `grunt.example.com` to the `EXAMPLE.COM` realm and also add an entry to put all hosts in the `.example.com` domain in the `EXAMPLE.COM` realm. The `krb.realms` file would be updated as follows:

```
# cat krb.realms
grunt.example.com EXAMPLE.COM
.example.com EXAMPLE.COM
.berkeley.edu CS.BERKELEY.EDU
.MIT.EDU ATHENA.MIT.EDU
.mit.edu ATHENA.MIT.EDU
```

Again, the other realms do not need to be there. They are here as an example of how a machine may be made aware of multiple realms. You may wish to remove them to simplify things.

The first line puts the *specific* system into the named realm. The rest of the lines show how to default systems of a particular subdomain to a named realm.

Now we are ready to create the database. This only needs to run on the Kerberos server (or Key Distribution Center). Issue the `kdb_init` command to do this:

```
# kdb_init
Realm name [default  ATHENA.MIT.EDU ]: EXAMPLE.COM
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
```

Enter Kerberos master key:

Now we have to save the key so that servers on the local machine can pick it up. Use the `kstash` command to do this:

```
# kstash
```

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!

This saves the encrypted master password in /etc/kerberosIV/master_key.

14.7.3. Making It All Run

Two principals need to be added to the database for *each* system that will be secured with Kerberos. Their names are `passwd` and `rcmd`. These two principals are made for each system, with the instance being the name of the individual system.

These daemons, **passwd** and **rcmd** allow other systems to change Kerberos passwords and run commands like `rcp(1)`, `rlogin(1)` and `rsh(1)`.

Now let us add these entries:

```
# kdb_edit
Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: passwd
Instance: grunt

<Not found>, Create [y] ? y

Principal: passwd, Instance: grunt, kdc_key_ver: 1
New Password:          <---- enter RANDOM here
Verifying password

New Password: <---- enter RANDOM here

Random password [y] ? y

Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name: rcmd
Instance: grunt

<Not found>, Create [y] ?

Principal: rcmd, Instance: grunt, kdc_key_ver: 1
New Password:          <---- enter RANDOM here
```

Verifying password

New Password: <---- enter RANDOM here

Random password [y] ?

Principal's new key version = 1

Expiration date (enter yyyy-mm-dd) [2000-01-01] ?

Max ticket lifetime (*5 minutes) [255] ?

Attributes [0] ?

Edit O.K.

Principal name: <---- null entry here will cause an exit

14.7.4. Creating the Server File

We now have to extract all the instances which define the services on each machine. For this we use the `ext_srvtab` command. This will create a file which must be copied or moved *by secure means* to each Kerberos client's `/etc/kerberosIV` directory. This file must be present on each server and client, and is crucial to the operation of Kerberos.

```
# ext_srvtab grunt
```

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered. BEWARE!

Generating 'grunt-new-srvtab'....

Now, this command only generates a temporary file which must be renamed to `srvtab` so that all the servers can pick it up. Use the `mv(1)` command to move it into place on the original system:

```
# mv grunt-new-srvtab srvtab
```

If the file is for a client system, and the network is not deemed safe, then copy the `client-new-srvtab` to removable media and transport it by secure physical means. Be sure to rename it to `srvtab` in the client's `/etc/kerberosIV` directory, and make sure it is mode 600:

```
# mv grumble-new-srvtab srvtab
```

```
# chmod 600 srvtab
```

14.7.5. Populating the Database

We now have to add some user entries into the database. First let us create an entry for the user `jane`. Use the `kdb_edit` command to do this:

```
# kdb_edit
```

Opening database...

Enter Kerberos master key:

```

Current Kerberos master key version is 1.

Master key entered.  BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: jane
Instance:

<Not found>, Create [y] ? y

Principal: jane, Instance: , kdc_key_ver: 1
New Password:          <---- enter a secure password here
Verifying password

New Password:          <---- re-enter the password here
Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:          <---- null entry here will cause an exit

```

14.7.6. Testing It All Out

First we have to start the Kerberos daemons. Note that if you have correctly edited your `/etc/rc.conf` then this will happen automatically when you reboot. This is only necessary on the Kerberos server. Kerberos clients will automatically get what they need from the `/etc/kerberosIV` directory.

```

# kerberos &
Kerberos server starting
Sleep forever on error
Log file is /var/log/kerberos.log
Current Kerberos master key version is 1.

Master key entered.  BEWARE!

Current Kerberos master key version is 1
Local realm: EXAMPLE.COM
# kadmind -n &
KADM Server KADM0.0A initializing
Please do not use 'kill -9' to kill this job, use a
regular kill instead

Current Kerberos master key version is 1.

Master key entered.  BEWARE!

```

Now we can try using the `kinit` command to get a ticket for the ID `jane` that we created above:

```

% kinit jane
MIT Project Athena (grunt.example.com)

```

```
Kerberos Initialization for "jane"
Password:
```

Try listing the tokens using `klist` to see if we really have them:

```
% klist
Ticket file:      /tmp/tkt245
Principal:        jane@EXAMPLE.COM

    Issued                Expires                Principal
Apr 30 11:23:22  Apr 30 19:23:22  krbtgt.EXAMPLE.COM@EXAMPLE.COM
```

Now try changing the password using `passwd(1)` to check if the **kpasswd** daemon can get authorization to the Kerberos database:

```
% passwd
realm EXAMPLE.COM
Old password for jane:
New Password for jane:
Verifying password
New Password for jane:
Password changed.
```

14.7.7. Adding `su` Privileges

Kerberos allows us to give *each* user who needs `root` privileges their own *separate* `su(1)` password. We could now add an ID which is authorized to `su(1)` to `root`. This is controlled by having an instance of `root` associated with a principal. Using `kdb_edit` we can create the entry `jane.root` in the Kerberos database:

```
# kdb_edit
Opening database...

Enter Kerberos master key:

Current Kerberos master key version is 1.

Master key entered.  BEWARE!
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: jane
Instance: root

<Not found>, Create [y] ? y

Principal: jane, Instance: root, kdc_key_ver: 1
New Password:          <---- enter a SECURE password here
Verifying password

New Password:          <---- re-enter the password here

Principal's new key version = 1
```

```

Expiration date (enter yyyy-mm-dd) [ 2000-01-01 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ? 12 <--- Keep this short!
Attributes [ 0 ] ?
Edit O.K.
Principal name: <----- null entry here will cause an exit

```

Now try getting tokens for it to make sure it works:

```

# kinit jane.root
MIT Project Athena (grunt.example.com)
Kerberos Initialization for "jane.root"
Password:

```

Now we need to add the user to root's .klogin file:

```

# cat /root/.klogin
jane.root@EXAMPLE.COM

```

Now try doing the su(1):

```

% su
Password:

```

and take a look at what tokens we have:

```

# klist
Ticket file:      /tmp/tkt_root_245
Principal:        jane.root@EXAMPLE.COM

    Issued                Expires                Principal
May  2 20:43:12  May  3 04:43:12  krbtgt.EXAMPLE.COM@EXAMPLE.COM

```

14.7.8. Using Other Commands

In an earlier example, we created a principal called `jane` with an instance `root`. This was based on a user with the same name as the principal, and this is a Kerberos default; that a `<principal>.<instance>` of the form `<username>.root` will allow that `<username>` to `su(1)` to `root` if the necessary entries are in the `.klogin` file in `root`'s home directory:

```

# cat /root/.klogin
jane.root@EXAMPLE.COM

```

Likewise, if a user has in their own home directory lines of the form:

```

% cat ~/.klogin
jane@EXAMPLE.COM
jack@EXAMPLE.COM

```

This allows anyone in the `EXAMPLE.COM` realm who has authenticated themselves as `jane` or `jack` (via `kinit`, see above) to access to `jane`'s account or files on this system (`grunt`) via `rlogin(1)`, `rsh(1)` or `rcp(1)`.

For example, `jane` now logs into another system using Kerberos:

```
% kinit
MIT Project Athena (grunt.example.com)
Password:
% rlogin grunt
Last login: Mon May  1 21:14:47 from grumble
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California.  All rights reserved.

FreeBSD BUILT-19950429 (GR386) #0: Sat Apr 29 17:50:09 SAT 1995
```

Or jack logs into jane's account on the same machine (jane having set up the .klogin file as above, and the person in charge of Kerberos having set up principal *jack* with a null instance):

```
% kinit
% rlogin grunt -l jane
MIT Project Athena (grunt.example.com)
Password:
Last login: Mon May  1 21:16:55 from grumble
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California.  All rights reserved.

FreeBSD BUILT-19950429 (GR386) #0: Sat Apr 29 17:50:09 SAT 1995
```

14.8. Kerberos5

Contributed by Tillman Hodgson. Based on a contribution by Mark Murray.

Every FreeBSD release beyond FreeBSD-5.1 includes support only for **Kerberos5**. Hence **Kerberos5** is the only version included, and its configuration is similar in many aspects to that of **KerberosIV**. The following information only applies to **Kerberos5** in post FreeBSD-5.0 releases. Users who wish to use the **KerberosIV** package may install the `security/krb4` port.

Kerberos is a network add-on system/protocol that allows users to authenticate themselves through the services of a secure server. Services such as remote login, remote copy, secure inter-system file copying and other high-risk tasks are made considerably safer and more controllable.

Kerberos can be described as an identity-verifying proxy system. It can also be described as a trusted third-party authentication system. **Kerberos** provides only one function — the secure authentication of users on the network. It does not provide authorization functions (what users are allowed to do) or auditing functions (what those users did). After a client and server have used **Kerberos** to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Therefore it is highly recommended that **Kerberos** be used with other security methods which provide authorization and audit services.

The following instructions can be used as a guide on how to set up **Kerberos** as distributed for FreeBSD. However, you should refer to the relevant manual pages for a complete description.

For purposes of demonstrating a **Kerberos** installation, the various name spaces will be handled as follows:

- The DNS domain („zone”) will be `example.org`.
- The **Kerberos** realm will be `EXAMPLE.ORG`.

Notatka: Please use real domain names when setting up **Kerberos** even if you intend to run it internally. This avoids DNS problems and assures inter-operation with other **Kerberos** realms.

14.8.1. History

Kerberos was created by MIT as a solution to network security problems. The **Kerberos** protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection.

Kerberos is both the name of a network authentication protocol and an adjective to describe programs that implement the program (**Kerberos** telnet, for example). The current version of the protocol is version 5, described in RFC 1510.

Several free implementations of this protocol are available, covering a wide range of operating systems. The Massachusetts Institute of Technology (MIT), where **Kerberos** was originally developed, continues to develop their **Kerberos** package. It is commonly used in the US as a cryptography product, as such it has historically been affected by US export regulations. The MIT **Kerberos** is available as a port (`security/krb5`). Heimdal **Kerberos** is another version 5 implementation, and was explicitly developed outside of the US to avoid export regulations (and is thus often included in non-commercial UNIX variants). The Heimdal **Kerberos** distribution is available as a port (`security/heimdal`), and a minimal installation of it is included in the base FreeBSD install.

In order to reach the widest audience, these instructions assume the use of the Heimdal distribution included in FreeBSD.

14.8.2. Setting up a Heimdal KDC

The Key Distribution Center (KDC) is the centralized authentication service that **Kerberos** provides — it is the computer that issues **Kerberos** tickets. The KDC is considered „trusted” by all other computers in the **Kerberos** realm, and thus has heightened security concerns.

Note that while running the **Kerberos** server requires very few computing resources, a dedicated machine acting only as a KDC is recommended for security reasons.

To begin setting up a KDC, ensure that your `/etc/rc.conf` file contains the correct settings to act as a KDC (you may need to adjust paths to reflect your own system):

```
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

Next we will set up your **Kerberos** config file, `/etc/krb5.conf`:

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
        kdc = kerberos.example.org
        admin_server = kerberos.example.org
    }
[domain_realm]
    .example.org = EXAMPLE.ORG
```

Note that this `/etc/krb5.conf` file implies that your KDC will have the fully-qualified hostname of `kerberos.example.org`. You will need to add a CNAME (alias) entry to your zone file to accomplish this if your KDC has a different hostname.

Notatka: For large networks with a properly configured BIND DNS server, the above example could be trimmed to:

```
[libdefaults]
    default_realm = EXAMPLE.ORG
```

With the following lines being appended to the `example.org` zonefile:

```
_kerberos._udp      IN  SRV      01 00 88 kerberos.example.org.
_kerberos._tcp      IN  SRV      01 00 88 kerberos.example.org.
_kpasswd._udp       IN  SRV      01 00 464 kerberos.example.org.
_kerberos-adm._tcp  IN  SRV      01 00 749 kerberos.example.org.
_kerberos           IN  TXT      EXAMPLE.ORG
```

Notatka: For clients to be able to find the **Kerberos** services, you *must* have either a fully configured `/etc/krb5.conf` or a minimally configured `/etc/krb5.conf` *and* a properly configured DNS server.

Next we will create the **Kerberos** database. This database contains the keys of all principals encrypted with a master password. You are not required to remember this password, it will be stored in a file (`/var/heimdal/m-key`). To create the master key, run `kstash` and enter a password.

Once the master key has been created, you can initialize the database using the `kadmin` program with the `-l` option (standing for „local”). This option instructs `kadmin` to modify the database files directly rather than going through the `kadmind` network service. This handles the chicken-and-egg problem of trying to connect to the database before it is created. Once you have the `kadmin` prompt, use the `init` command to create your realms initial database.

Lastly, while still in `kadmin`, create your first principal using the `add` command. Stick to the defaults options for the principal for now, you can always change them later with the `modify` command. Note that you can use the `?` command at any prompt to see the available options.

A sample database creation session is shown below:

```
# kstash
Master key: xxxxxxxx
Verifying password - Master key: xxxxxxxx

# kadmin -l
kadmin> init EXAMPLE.ORG
Realm max ticket life [unlimited]:
kadmin> add tillman
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
Password: xxxxxxxx
Verifying password - Password: xxxxxxxx
```

Now it is time to start up the KDC services. Run `/etc/rc.d/kerberos start` and `/etc/rc.d/kadmind start` to bring up the services. Note that you will not have any kerberized daemons running at this point but you should be able to confirm that the KDC is functioning by obtaining and listing a ticket for the principal (user) that you just created from the command-line of the KDC itself:

```
% k5init tillman
tillman@EXAMPLE.ORG's Password:

% k5list
Credentials cache: FILE:/tmp/krb5cc_500
Principal: tillman@EXAMPLE.ORG

Issued                Expires                Principal
Aug 27 15:37:58      Aug 28 01:37:58      krbtgt/EXAMPLE.ORG@EXAMPLE.ORG
```

14.8.3. Kerberos enabling a server with Heimdal services

First, we need a copy of the **Kerberos** configuration file, `/etc/krb5.conf`. To do so, simply copy it over to the client computer from the KDC in a secure fashion (using network utilities, such as `scp(1)`, or physically via a floppy disk).

Next you need a `/etc/krb5.keytab` file. This is the major difference between a server providing **Kerberos** enabled daemons and a workstation — the server must have a `keytab` file. This file contains the servers host key, which allows it and the KDC to verify each others identity. It must be transmitted to the server in a secure fashion, as the security of the server can be broken if the key is made public. This explicitly means that transferring it via a clear text channel, such as FTP, is a very bad idea.

Typically, you transfer the `keytab` to the server using the `kadmin` program. This is handy because you also need to create the host principal (the KDC end of the `krb5.keytab`) using `kadmin`.

Note that you must have already obtained a ticket and that this ticket must be allowed to use the `kadmin` interface in the `kadmind.acl`. See the section titled „Remote administration” in the Heimdal info pages (`info heimdal`) for details on designing access control lists. If you do not want to enable remote `kadmin` access, you can simply securely connect to the KDC (via local console, `ssh(1)` or **Kerberos** `telnet(1)`) and perform administration locally using `kadmin -l`.

After installing the `/etc/krb5.conf` file, you can use `kadmin` from the **Kerberos** server. The `add --random-key` command will let you add the servers host principal, and the `ext` command will allow you to extract the servers host principal to its own `keytab`. For example:

```
# kadmin
kadmin> add --random-key host/myserver.example.org
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
kadmin> ext host/myserver.example.org
kadmin> exit
```

Note that the `ext` command (short for „extract”) stores the extracted key in `/etc/krb5.keytab` by default.

If you do not have `kadmind` running on the KDC (possibly for security reasons) and thus do not have access to `kadmin` remotely, you can add the host principal (`host/myserver.EXAMPLE.ORG`) directly on the KDC and then extract it to a temporary file (to avoid over-writing the `/etc/krb5.keytab` on the KDC) using something like this:

```
# kadmin
kadmin> ext --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

You can then securely copy the keytab to the server computer (using `scp` or a floppy, for example). Be sure to specify a non-default keytab name to avoid over-writing the keytab on the KDC.

At this point your server can communicate with the KDC (due to its `krb5.conf` file) and it can prove its own identity (due to the `krb5.keytab` file). It is now ready for you to enable some **Kerberos** services. For this example we will enable the `telnet` service by putting a line like this into your `/etc/inetd.conf` and then restarting the `inetd(8)` service with `/etc/rc.d/inetd restart`:

```
telnet      stream  tcp      nowait  root    /usr/libexec/telnetd  telnetd -a user
```

The critical bit is that the `-a` (for authentication) type is set to `user`. Consult the `telnetd(8)` manual page for more details.

14.8.4. Kerberos enabling a client with Heimdal

Setting up a client computer is almost trivially easy. As far as **Kerberos** configuration goes, you only need the **Kerberos** configuration file, located at `/etc/krb5.conf`. Simply securely copy it over to the client computer from the KDC.

Test your client computer by attempting to use `kinit`, `klist`, and `kdestroy` from the client to obtain, show, and then delete a ticket for the principal you created above. You should also be able to use **Kerberos** applications to connect to **Kerberos** enabled servers, though if that does not work and obtaining a ticket does the problem is likely with the server and not with the client or the KDC.

When testing an application like `telnet`, try using a packet sniffer (such as `tcpdump(1)`) to confirm that your password is not sent in the clear. Try using `telnet` with the `-x` option, which encrypts the entire data stream (similar to `ssh`).

The core **Kerberos** client applications (traditionally named `kinit`, `klist`, `kdestroy`, and `kpasswd`) are installed in the base FreeBSD install. Note that FreeBSD versions prior to 5.0 renamed them to `k5init`, `k5list`, `k5destroy`, `k5passwd`, and `k5stash` (though it is typically only used once).

Various non-core **Kerberos** client applications are also installed by default. This is where the „minimal” nature of the base Heimdal installation is felt: `telnet` is the only **Kerberos** enabled service.

The Heimdal port adds some of the missing client applications: **Kerberos** enabled versions of `ftp`, `rsh`, `rcp`, `rlogin`, and a few other less common programs. The MIT port also contains a full suite of **Kerberos** client applications.

14.8.5. User configuration files: `.k5login` and `.k5users`

Users within a realm typically have their **Kerberos** principal (such as `tillman@EXAMPLE.ORG`) mapped to a local user account (such as a local account named `tillman`). Client applications such as `telnet` usually do not require a user name or a principal.

Occasionally, however, you want to grant access to a local user account to someone who does not have a matching **Kerberos** principal. For example, `tillman@EXAMPLE.ORG` may need access to the local user account `webdevelopers`. Other principals may also need access to that local account.

The `.k5login` and `.k5users` files, placed in a user's home directory, can be used similar to a powerful combination of `.hosts` and `.rhosts`, solving this problem. For example, if a `.k5login` with the following contents:

```
tillman@example.org
jdoe@example.org
```

Were to be placed into the home directory of the local user `webdevelopers` then both principals listed would have access to that account without requiring a shared password.

Reading the manual pages for these commands is recommended. Note that the `ksu` manual page covers `.k5users`.

14.8.6. Kerberos Tips, Tricks, and Troubleshooting

- When using either the Heimdal or MIT **Kerberos** ports ensure that your `PATH` environment variable lists the **Kerberos** versions of the client applications before the system versions.
- Do all the computers in your realm have synchronized time settings? If not, authentication may fail. Sekcja 25.10 describes how to synchronize clocks using NTP.
- MIT and Heimdal inter-operate nicely. Except for `kadmin`, the protocol for which is not standardized.
- If you change your hostname, you also need to change your `host/` principal and update your keytab. This also applies to special keytab entries like the `www/` principal used for Apache's `www/mod_auth_kerb`.
- All hosts in your realm must be resolvable (both forwards and reverse) in DNS (or `/etc/hosts` as a minimum). CNAMEs will work, but the A and PTR records must be correct and in place. The error message is not very intuitive: `Kerberos5 refuses authentication because Read req failed: Key table entry not found`.
- Some operating systems that may be acting as clients to your KDC do not set the permissions for `ksu` to be `setuid root`. This means that `ksu` does not work, which is a good security idea but annoying. This is not a KDC error.
- With MIT **Kerberos**, if you want to allow a principal to have a ticket life longer than the default ten hours, you must use `modify_principal` in `kadmin` to change the `maxlife` of both the principal in question and the `krbtgt` principal. Then the principal can use the `-l` option with `kinit` to request a ticket with a longer lifetime.
-

Notatka: If you run a packet sniffer on your KDC to add in troubleshooting and then run `kinit` from a workstation, you will notice that your TGT is sent immediately upon running `kinit` — even before you type your password! The explanation is that the **Kerberos** server freely transmits a TGT (Ticket Granting Ticket) to any unauthorized request; however, every TGT is encrypted in a key derived from the user's password. Therefore, when a user types their password it is not being sent to the KDC, it is being used to decrypt the TGT that `kinit` already obtained. If the decryption process results in a valid ticket with a valid time stamp, the user has valid **Kerberos** credentials. These credentials include a session key for establishing secure communications with the **Kerberos** server in the future, as well as the actual ticket-granting ticket, which is actually encrypted with the **Kerberos** server's own key. This second layer of encryption is unknown to the user, but it is what allows the **Kerberos** server to verify the authenticity of each TGT.

- If you want to use long ticket lifetimes (a week, for example) and you are using **OpenSSH** to connect to the machine where your ticket is stored, make sure that **Kerberos** `TicketCleanup` is set to `no` in your `sshd_config` or else your tickets will be deleted when you log out.
- Remember that host principals can have a longer ticket lifetime as well. If your user principal has a lifetime of a week but the host you are connecting to has a lifetime of nine hours, you will have an expired host principal in your cache and the ticket cache will not work as expected.
- When setting up a `krb5.dict` file to prevent specific bad passwords from being used (the manual page for `kadmin` covers this briefly), remember that it only applies to principals that have a password policy assigned to them. The `krb5.dict` files format is simple: one string per line. Creating a symbolic link to `/usr/share/dict/words` might be useful.

14.8.7. Differences with the MIT port

The major difference between the MIT and Heimdal installs relates to the `kadmin` program which has a different (but equivalent) set of commands and uses a different protocol. This has a large implications if your KDC is MIT as you will not be able to use the Heimdal `kadmin` program to administer your KDC remotely (or vice versa, for that matter).

The client applications may also take slightly different command line options to accomplish the same tasks. Following the instructions on the MIT **Kerberos** web site (<http://web.mit.edu/Kerberos/www/>) is recommended. Be careful of path issues: the MIT port installs into `/usr/local/` by default, and the „normal” system applications may be run instead of MIT if your `PATH` environment variable lists the system directories first.

Notatka: With the MIT `security/krb5` port that is provided by FreeBSD, be sure to read the `/usr/local/share/doc/krb5/README.FreeBSD` file installed by the port if you want to understand why logins via `telnetd` and `klogind` behave somewhat oddly. Most importantly, correcting the „incorrect permissions on cache file” behavior requires that the `login.krb5` binary be used for authentication so that it can properly change ownership for the forwarded credentials.

14.8.8. Mitigating limitations found in Kerberos

14.8.8.1. Kerberos is an all-or-nothing approach

Every service enabled on the network must be modified to work with **Kerberos** (or be otherwise secured against network attacks) or else the users credentials could be stolen and re-used. An example of this would be **Kerberos** enabling all remote shells (via `rsh` and `telnet`, for example) but not converting the POP3 mail server which sends passwords in plain text.

14.8.8.2. Kerberos is intended for single-user workstations

In a multi-user environment, **Kerberos** is less secure. This is because it stores the tickets in the `/tmp` directory, which is readable by all users. If a user is sharing a computer with several other people simultaneously (i.e. multi-user), it is possible that the user's tickets can be stolen (copied) by another user.

This can be overcome with the `-c filename` command-line option or (preferably) the `KRB5CCNAME` environment variable, but this is rarely done. In principal, storing the ticket in the users home directory and using simple file permissions can mitigate this problem.

14.8.8.3. The KDC is a single point of failure

By design, the KDC must be as secure as the master password database is contained on it. The KDC should have absolutely no other services running on it and should be physically secured. The danger is high because **Kerberos** stores all passwords encrypted with the same key (the „master” key), which in turn is stored as a file on the KDC.

As a side note, a compromised master key is not quite as bad as one might normally fear. The master key is only used to encrypt the **Kerberos** database and as a seed for the random number generator. As long as access to your KDC is secure, an attacker cannot do much with the master key.

Additionally, if the KDC is unavailable (perhaps due to a denial of service attack or network problems) the network services are unusable as authentication can not be performed, a recipe for a denial-of-service attack. This can be alleviated with multiple KDCs (a single master and one or more slaves) and with careful implementation of secondary or fall-back authentication (PAM is excellent for this).

14.8.8.4. Kerberos Shortcomings

Kerberos allows users, hosts and services to authenticate between themselves. It does not have a mechanism to authenticate the KDC to the users, hosts or services. This means that a trojanned `kinit` (for example) could record all user names and passwords. Something like `security/tripwire` or other file system integrity checking tools can alleviate this.

14.8.9. Resources and further information

- The **Kerberos** FAQ (<http://www.faqs.org/faqs/Kerberos-faq/general/preamble.html>)
- Designing an Authentication System: a Dialog in Four Scenes (<http://web.mit.edu/Kerberos/www/dialogue.html>)
- RFC 1510, The **Kerberos** Network Authentication Service (V5) (<http://www.ietf.org/rfc/rfc1510.txt?number=1510>)
- MIT **Kerberos** home page (<http://web.mit.edu/Kerberos/www/>)
- Heimdal **Kerberos** home page (<http://www.pdc.kth.se/heimdal/>)

14.9. OpenSSL

Written by: Tom Rhodes.

One feature that many users overlook is the **OpenSSL** toolkit included in FreeBSD. **OpenSSL** provides an encryption transport layer on top of the normal communications layer; thus allowing it to be intertwined with many network applications and services.

Some uses of **OpenSSL** may include encrypted authentication of mail clients, web based transactions such as credit card payments and more. Many ports such as `www/apache13-ssl`, and `mail/sylpheed-claws` will offer compilation support for building with **OpenSSL**.

Notatka: In most cases the Ports Collection will attempt to build the `security/openssl` port unless the `WITH_OPENSSL_BASE` make variable is explicitly set to „yes”.

The version of **OpenSSL** included in FreeBSD supports Secure Sockets Layer v2/v3 (SSLv2/SSLv3), Transport Layer Security v1 (TLSv1) network security protocols and can be used as a general cryptographic library.

Notatka: While **OpenSSL** supports the IDEA algorithm, it is disabled by default due to United States patents. To use it, the license should be reviewed and, if the restrictions are acceptable, the `MAKE_IDEA` variable must be set in `make.conf`.

One of the most common uses of **OpenSSL** is to provide certificates for use with software applications. These certificates ensure that the credentials of the company or individual are valid and not fraudulent. If the certificate in question has not been verified by one of the several „Certificate Authorities”, or CAs, a warning is usually produced. A Certificate Authority is a company, such as VeriSign (<http://www.verisign.com>), which will sign certificates in order to validate credentials of individuals or companies. This process has a cost associated with it and is definitely not a requirement for using certificates; however, it can put some of the more paranoid users at ease.

14.9.1. Generating Certificates

To generate a certificate, the following command is available:

```
# openssl req -new -nodes -out req.pem -keyout cert.pem
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'cert.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:PA
Locality Name (eg, city) []:Pittsburgh
```

```

Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company
Organizational Unit Name (eg, section) []:Systems Administrator
Common Name (eg, YOUR name) []:localhost.example.org
Email Address []:trhodes@FreeBSD.org

```

```

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:SOME PASSWORD
An optional company name []:Another Name

```

Notice the response directly after the „Common Name” prompt shows a domain name. This prompt requires a server name to be entered for verification purposes; placing anything but a domain name would yield a useless certificate. Other options, for instance expire time, alternate encryption algorithms, etc. are available. A complete list may be obtained by viewing the openssl(1) manual page.

Two files should now exist in the directory in which the aforementioned command was issued. The certificate request, `req.pem`, may be sent to a certificate authority who will validate the credentials that you entered, sign the request and return the certificate to you. The second file created will be named `cert.pem` and is the private key for the certificate and should be protected at all costs; if this falls in the hands of others it can be used to impersonate you (or your server).

In cases where a signature from a CA is not required, a self signed certificate can be created. First, generate the RSA key:

```
# openssl dsaparam -rand -genkey -out myRSA.key 1024
```

Next, generate the CA key:

```
# openssl genrsa -des3 -out myca.key myRSA.key
```

Use this key to create the certificate:

```
# openssl req -new -x509 -days 365 -key myca.key -out new.crt
```

Two new files should appear in the directory: a certificate authority signature file, `myca.key` and the certificate itself, `new.crt`. These should be placed in a directory, preferably under `/etc`, which is readable only by `root`. Permissions of 0700 should be fine for this and they can be set with the `chmod` utility.

14.9.2. Using Certificates, an Example

So what can these files do? A good use would be to encrypt connections to the **Sendmail** MTA. This would dissolve the use of clear text authentication for users who send mail via the local MTA.

Notatka: This is not the best use in the world as some MUAs will present the user with an error if they have not installed the certificate locally. Refer to the documentation included with the software for more information on certificate installation.

The following lines should be placed inside the local `.mc` file:

```

dnl SSL Options
define('confCACERT_PATH', '/etc/certs')dnl

```

```
define('confCACERT','/etc/certs/new.crt')dnl
define('confSERVER_CERT','/etc/certs/new.crt')dnl
define('confSERVER_KEY','/etc/certs/myca.key')dnl
define('confTLS_SRV_OPTIONS','V')dnl
```

Where `/etc/certs/` is the directory to be used for storing the certificate and key files locally. The last few requirements are a rebuild of the local `.cf` file. This is easily achieved by typing `make install` within the `/etc/mail` directory. Follow that up with `make restart` which should start the **Sendmail** daemon.

If all went well there will be no error messages in the `/var/log/maillog` file and **Sendmail** will show up in the process list.

For a simple test, simply connect to the mail server using the `telnet(1)` utility:

```
# telnet example.com 25
Trying 192.0.34.166...
Connected to example.com.
Escape character is '^]'.
220 example.com ESMTP Sendmail 8.12.10/8.12.10; Tue, 31 Aug 2004 03:41:22 -0400 (EDT)
ehlo example.com
250-example.com Hello example.com [192.0.34.166], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-STARTTLS
250-DELIVERBY
250 HELP
quit
221 2.0.0 example.com closing connection
Connection closed by foreign host.
```

If the „STARTTLS” line appears in the output then everything is working correctly.

14.10. VPN over IPsec

Written by Nik Clayton.

Creating a VPN between two networks, separated by the Internet, using FreeBSD gateways.

14.10.1. Understanding IPsec

Written by Hiten M. Pandya.

This section will guide you through the process of setting up IPsec, and to use it in an environment which consists of FreeBSD and **Microsoft Windows 2000/XP** machines, to make them communicate securely. In order to set up IPsec, it is necessary that you are familiar with the concepts of building a custom kernel (see Rozdział 8).

IPsec is a protocol which sits on top of the Internet Protocol (IP) layer. It allows two or more hosts to communicate in a secure manner (hence the name). The FreeBSD IPsec „network stack” is based on the KAME (<http://www.kame.net/>) implementation, which has support for both protocol families, IPv4 and IPv6.

Notatka: FreeBSD contains a „hardware accelerated” IPsec stack, known as „Fast IPsec”, that was obtained from OpenBSD. It employs cryptographic hardware (whenever possible) via the `crypto(4)` subsystem to optimize the performance of IPsec. This subsystem is new, and does not support all the features that are available in the KAME version of IPsec. However, in order to enable hardware-accelerated IPsec, the following kernel option has to be added to your kernel configuration file:

```
options    FAST_IPSEC    # new IPsec (cannot define w/ IPSEC)
```

Note, that it is not currently possible to use the „Fast IPsec” subsystem in lieu of the KAME implementation of IPsec. Consult the `fast_ipsec(4)` manual page for more information.

Notatka: To let firewalls properly track state for `gif(4)` tunnels too, you have to enable the `IPSEC_FILTERGIF` in your kernel configuration:

```
options    IPSEC_FILTERGIF    #filter ipsec packets from a tunnel
```

IPsec consists of two sub-protocols:

- *Encapsulated Security Payload (ESP)*, protects the IP packet data from third party interference, by encrypting the contents using symmetric cryptography algorithms (like Blowfish, 3DES).
- *Authentication Header (AH)*, protects the IP packet header from third party interference and spoofing, by computing a cryptographic checksum and hashing the IP packet header fields with a secure hashing function. This is then followed by an additional header that contains the hash, to allow the information in the packet to be authenticated.

ESP and AH can either be used together or separately, depending on the environment.

IPsec can either be used to directly encrypt the traffic between two hosts (known as *Transport Mode*); or to build „virtual tunnels” between two subnets, which could be used for secure communication between two corporate networks (known as *Tunnel Mode*). The latter is more commonly known as a *Virtual Private Network (VPN)*. The `ipsec(4)` manual page should be consulted for detailed information on the IPsec subsystem in FreeBSD.

To add IPsec support to your kernel, add the following options to your kernel configuration file:

```
options    IPSEC            #IP security
options    IPSEC_ESP        #IP security (crypto; define w/ IPSEC)
```

If IPsec debugging support is desired, the following kernel option should also be added:

```
options    IPSEC_DEBUG      #debug for IP security
```

14.10.2. The Problem

There is no standard for what constitutes a VPN. VPNs can be implemented using a number of different technologies, each of which have their own strengths and weaknesses. This section presents a scenario, and the strategies used for implementing a VPN for this scenario.

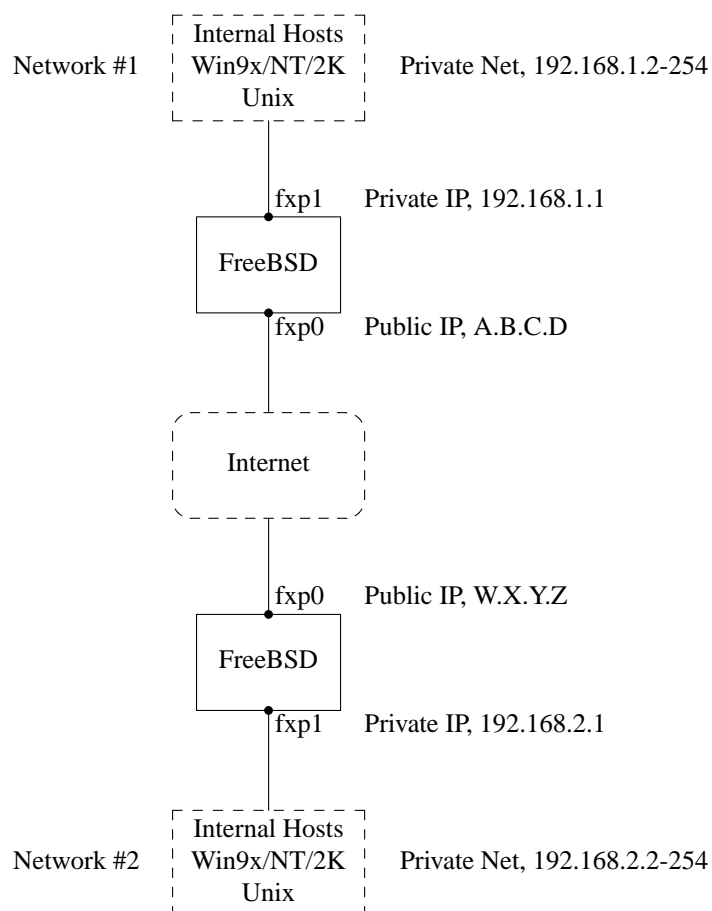
14.10.3. The Scenario: Two networks, connected to the Internet, to behave as one

The premise is as follows:

- You have at least two sites
- Both sites are using IP internally
- Both sites are connected to the Internet, through a gateway that is running FreeBSD.
- The gateway on each network has at least one public IP address.
- The internal addresses of the two networks can be public or private IP addresses, it does not matter. You can be running NAT on the gateway machine if necessary.
- The internal IP addresses of the two networks *do not collide*. While I expect it is theoretically possible to use a combination of VPN technology and NAT to get this to work, I expect it to be a configuration nightmare.

If you find that you are trying to connect two networks, both of which, internally, use the same private IP address range (e.g. both of them use 192.168.1.x), then one of the networks will have to be renumbered.

The network topology might look something like this:



Notice the two public IP addresses. I will use the letters to refer to them in the rest of this article. Anywhere you see those letters in this article, replace them with your own public IP addresses. Note also that internally, the two gateway machines have .1 IP addresses, and that the two networks have different private IP addresses (192.168.1.x and 192.168.2.x respectively). All the machines on the private networks have been configured to use the .1 machine as their default gateway.

The intention is that, from a network point of view, each network should view the machines on the other network as though they were directly attached the same router -- albeit a slightly slow router with an occasional tendency to drop packets.

This means that (for example), machine 192.168.1.20 should be able to run

```
ping 192.168.2.34
```

and have it work, transparently. Windows machines should be able to see the machines on the other network, browse file shares, and so on, in exactly the same way that they can browse machines on the local network.

And the whole thing has to be secure. This means that traffic between the two networks has to be encrypted.

Creating a VPN between these two networks is a multi-step process. The stages are as follows:

1. Create a „virtual” network link between the two networks, across the Internet. Test it, using tools like ping(8), to make sure it works.

2. Apply security policies to ensure that traffic between the two networks is transparently encrypted and decrypted as necessary. Test this, using tools like `tcpdump(1)`, to ensure that traffic is encrypted.
3. Configure additional software on the FreeBSD gateways, to allow Windows machines to see one another across the VPN.

14.10.3.1. Step 1: Creating and testing a „virtual” network link

Suppose that you were logged in to the gateway machine on network #1 (with public IP address `A.B.C.D`, private IP address `192.168.1.1`), and you ran `ping 192.168.2.1`, which is the private address of the machine with IP address `W.X.Y.Z`. What needs to happen in order for this to work?

1. The gateway machine needs to know how to reach `192.168.2.1`. In other words, it needs to have a route to `192.168.2.1`.
2. Private IP addresses, such as those in the `192.168.x` range are not supposed to appear on the Internet at large. Instead, each packet you send to `192.168.2.1` will need to be wrapped up inside another packet. This packet will need to appear to be from `A.B.C.D`, and it will have to be sent to `W.X.Y.Z`. This process is called *encapsulation*.
3. Once this packet arrives at `W.X.Y.Z` it will need to „unencapsulated”, and delivered to `192.168.2.1`.

You can think of this as requiring a „tunnel” between the two networks. The two „tunnel mouths” are the IP addresses `A.B.C.D` and `W.X.Y.Z`, and the tunnel must be told the addresses of the private IP addresses that will be allowed to pass through it. The tunnel is used to transfer traffic with private IP addresses across the public Internet.

This tunnel is created by using the generic interface, or `gif` devices on FreeBSD. As you can imagine, the `gif` interface on each gateway host must be configured with four IP addresses; two for the public IP addresses, and two for the private IP addresses.

Support for the `gif` device must be compiled in to the FreeBSD kernel on both machines. You can do this by adding the line:

```
device gif
```

to the kernel configuration files on both machines, and then compile, install, and reboot as normal.

Configuring the tunnel is a two step process. First the tunnel must be told what the outside (or public) IP addresses are, using `ifconfig(8)`. Then the private IP addresses must be configured using `ifconfig(8)`.

On the gateway machine on network #1 you would run the following two commands to configure the tunnel.

```
ifconfig gif0 A.B.C.D W.X.Y.Z
ifconfig gif0 inet 192.168.1.1 192.168.2.1 netmask 0xffffffff
```

On the other gateway machine you run the same commands, but with the order of the IP addresses reversed.

```
ifconfig gif0 W.X.Y.Z A.B.C.D
ifconfig gif0 inet 192.168.2.1 192.168.1.1 netmask 0xffffffff
```

You can then run:

```
ifconfig gif0
```

to see the configuration. For example, on the network #1 gateway, you would see this:

```
# ifconfig gif0
gif0: flags=8011<UP,POINTTOPOINT,MULTICAST> mtu 1280
inet 192.168.1.1 --> 192.168.2.1 netmask 0xffffffff
physical address inet A.B.C.D --> W.X.Y.Z
```

As you can see, a tunnel has been created between the physical addresses A.B.C.D and W.X.Y.Z, and the traffic allowed through the tunnel is that between 192.168.1.1 and 192.168.2.1.

This will also have added an entry to the routing table on both machines, which you can examine with the command `netstat -rn`. This output is from the gateway host on network #1.

```
# netstat -rn
Routing tables

Internet:
Destination      Gateway          Flags    Refs      Use    Netif    Expire
...
192.168.2.1       192.168.1.1     UH        0          0      gif0
...
```

As the „Flags” value indicates, this is a host route, which means that each gateway knows how to reach the other gateway, but they do not know how to reach the rest of their respective networks. That problem will be fixed shortly.

It is likely that you are running a firewall on both machines. This will need to be circumvented for your VPN traffic. You might want to allow all traffic between both networks, or you might want to include firewall rules that protect both ends of the VPN from one another.

It greatly simplifies testing if you configure the firewall to allow all traffic through the VPN. You can always tighten things up later. If you are using `ipfw(8)` on the gateway machines then a command like

```
ipfw add 1 allow ip from any to any via gif0
```

will allow all traffic between the two end points of the VPN, without affecting your other firewall rules. Obviously you will need to run this command on both gateway hosts.

This is sufficient to allow each gateway machine to ping the other. On 192.168.1.1, you should be able to run

```
ping 192.168.2.1
```

and get a response, and you should be able to do the same thing on the other gateway machine.

However, you will not be able to reach internal machines on either network yet. This is because of the routing -- although the gateway machines know how to reach one another, they do not know how to reach the network behind each one.

To solve this problem you must add a static route on each gateway machine. The command to do this on the first gateway would be:

```
route add 192.168.2.0 192.168.2.1 netmask 0xffffffff00
```

This says „In order to reach the hosts on the network 192.168.2.0, send the packets to the host 192.168.2.1”. You will need to run a similar command on the other gateway, but with the 192.168.1.x addresses instead.

IP traffic from hosts on one network will now be able to reach hosts on the other network.

That has now created two thirds of a VPN between the two networks, in as much as it is „virtual” and it is a „network”. It is not private yet. You can test this using ping(8) and tcpdump(1). Log in to the gateway host and run

```
tcpdump dst host 192.168.2.1
```

In another log in session on the same host run

```
ping 192.168.2.1
```

You will see output that looks something like this:

```
16:10:24.018080 192.168.1.1 > 192.168.2.1: icmp: echo request
16:10:24.018109 192.168.1.1 > 192.168.2.1: icmp: echo reply
16:10:25.018814 192.168.1.1 > 192.168.2.1: icmp: echo request
16:10:25.018847 192.168.1.1 > 192.168.2.1: icmp: echo reply
16:10:26.028896 192.168.1.1 > 192.168.2.1: icmp: echo request
16:10:26.029112 192.168.1.1 > 192.168.2.1: icmp: echo reply
```

As you can see, the ICMP messages are going back and forth unencrypted. If you had used the -s parameter to tcpdump(1) to grab more bytes of data from the packets you would see more information.

Obviously this is unacceptable. The next section will discuss securing the link between the two networks so that it all traffic is automatically encrypted.

Summary:

- Configure both kernels with „device gif”.
- Edit /etc/rc.conf on gateway host #1 and add the following lines (replacing IP addresses as necessary).

```
gifconfig_gif0="A.B.C.D W.X.Y.Z"
ifconfig_gif0="inet 192.168.1.1 192.168.2.1 netmask 0xffffffff"
static_routes="vpn"
route_vpn="192.168.2.0 192.168.2.1 netmask 0xfffff00"
```

- Edit your firewall script (/etc/rc.firewall, or similar) on both hosts, and add
ipfw add 1 allow ip from any to any via gif0
- Make similar changes to /etc/rc.conf on gateway host #2, reversing the order of IP addresses.

14.10.3.2. Step 2: Securing the link

To secure the link we will be using IPsec. IPsec provides a mechanism for two hosts to agree on an encryption key, and to then use this key in order to encrypt data between the two hosts.

There are two areas of configuration to be considered here.

1. There must be a mechanism for two hosts to agree on the encryption mechanism to use. Once two hosts have agreed on this mechanism there is said to be a „security association” between them.

2. There must be a mechanism for specifying which traffic should be encrypted. Obviously, you do not want to encrypt all your outgoing traffic -- you only want to encrypt the traffic that is part of the VPN. The rules that you put in place to determine what traffic will be encrypted are called „security policies”.

Security associations and security policies are both maintained by the kernel, and can be modified by userland programs. However, before you can do this you must configure the kernel to support IPsec and the Encapsulated Security Payload (ESP) protocol. This is done by configuring a kernel with:

```
options IPSEC
options IPSEC_ESP
```

and recompiling, reinstalling, and rebooting. As before you will need to do this to the kernels on both of the gateway hosts.

You have two choices when it comes to setting up security associations. You can configure them by hand between two hosts, which entails choosing the encryption algorithm, encryption keys, and so forth, or you can use daemons that implement the Internet Key Exchange protocol (IKE) to do this for you.

I recommend the latter. Apart from anything else, it is easier to set up.

Editing and displaying security policies is carried out using `setkey(8)`. By analogy, `setkey` is to the kernel's security policy tables as `route(8)` is to the kernel's routing tables. `setkey` can also display the current security associations, and to continue the analogy further, is akin to `netstat -r` in that respect.

There are a number of choices for daemons to manage security associations with FreeBSD. This article will describe how to use one of these, `racoon` — which is available from `security/ipsec-tools` in the FreeBSD Ports collection.

The **racoon** software must be run on both gateway hosts. On each host it is configured with the IP address of the other end of the VPN, and a secret key (which you choose, and must be the same on both gateways).

The two daemons then contact one another, confirm that they are who they say they are (by using the secret key that you configured). The daemons then generate a new secret key, and use this to encrypt the traffic over the VPN. They periodically change this secret, so that even if an attacker were to crack one of the keys (which is as theoretically close to unfeasible as it gets) it will not do them much good -- by the time they have cracked the key the two daemons have chosen another one.

The configuration file for `racoon` is stored in `${PREFIX}/etc/racoon`. You should find a configuration file there, which should not need to be changed too much. The other component of `racoon`'s configuration, which you will need to change, is the „pre-shared key”.

The default `racoon` configuration expects to find this in the file `${PREFIX}/etc/racoon/psk.txt`. It is important to note that the pre-shared key is *not* the key that will be used to encrypt your traffic across the VPN link, it is simply a token that allows the key management daemons to trust one another.

`psk.txt` contains a line for each remote site you are dealing with. In this example, where there are two sites, each `psk.txt` file will contain one line (because each end of the VPN is only dealing with one other end).

On gateway host #1 this line should look like this:

```
W.X.Y.Z          secret
```

That is, the *public* IP address of the remote end, whitespace, and a text string that provides the secret. Obviously, you should not use „secret” as your key -- the normal rules for choosing a password apply.

On gateway host #2 the line would look like this

```
A.B.C.D          secret
```

That is, the public IP address of the remote end, and the same secret key. `psk.txt` must be mode `0600` (i.e., only read/write to `root`) before `racoon` will run.

You must run `racoon` on both gateway machines. You will also need to add some firewall rules to allow the IKE traffic, which is carried over UDP to the ISAKMP (Internet Security Association Key Management Protocol) port. Again, this should be fairly early in your firewall ruleset.

```
ipfw add 1 allow udp from A.B.C.D to W.X.Y.Z isakmp
ipfw add 1 allow udp from W.X.Y.Z to A.B.C.D isakmp
```

Once `racoon` is running you can try pinging one gateway host from the other. The connection is still not encrypted, but `racoon` will then set up the security associations between the two hosts -- this might take a moment, and you may see this as a short delay before the ping commands start responding.

Once the security association has been set up you can view it using `setkey(8)`. Run

```
setkey -D
```

on either host to view the security association information.

That's one half of the problem. The other half is setting your security policies.

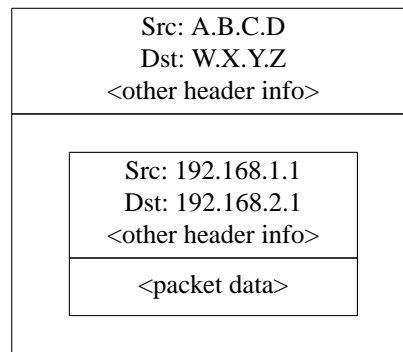
To create a sensible security policy, let's review what's been set up so far. This discussion holds for both ends of the link.

Each IP packet that you send out has a header that contains data about the packet. The header includes the IP addresses of both the source and destination. As we already know, private IP addresses, such as the `192.168.x.y` range are not supposed to appear on the public Internet. Instead, they must first be encapsulated inside another packet. This packet must have the public source and destination IP addresses substituted for the private addresses.

So if your outgoing packet started looking like this:

Src: 192.168.1.1 Dst: 192.168.2.1 <other header info>
<packet data>

Then it will be encapsulated inside another packet, looking something like this:



This encapsulation is carried out by the `gif` device. As you can see, the packet now has real IP addresses on the outside, and our original packet has been wrapped up as data inside the packet that will be put out on the Internet.

Obviously, we want all traffic between the VPNs to be encrypted. You might try putting this in to words, as:

„If a packet leaves from `A.B.C.D`, and it is destined for `W.X.Y.Z`, then encrypt it, using the necessary security associations.”

„If a packet arrives from `W.X.Y.Z`, and it is destined for `A.B.C.D`, then decrypt it, using the necessary security associations.”

That’s close, but not quite right. If you did this, all traffic to and from `W.X.Y.Z`, even traffic that was not part of the VPN, would be encrypted. That’s not quite what you want. The correct policy is as follows

„If a packet leaves from `A.B.C.D`, and that packet is encapsulating another packet, and it is destined for `W.X.Y.Z`, then encrypt it, using the necessary security associations.”

„If a packet arrives from `W.X.Y.Z`, and that packet is encapsulating another packet, and it is destined for `A.B.C.D`, then decrypt it, using the necessary security associations.”

A subtle change, but a necessary one.

Security policies are also set using `setkey(8)`. `setkey(8)` features a configuration language for defining the policy. You can either enter configuration instructions via `stdin`, or you can use the `-f` option to specify a filename that contains configuration instructions.

The configuration on gateway host #1 (which has the public IP address `A.B.C.D`) to force all outbound traffic to `W.X.Y.Z` to be encrypted is:

```
spdadd A.B.C.D/32 W.X.Y.Z/32 ipencap -P out ipsec esp/tunnel/A.B.C.D-W.X.Y.Z/require;
```

Put these commands in a file (e.g. `/etc/ipsec.conf`) and then run

```
# setkey -f /etc/ipsec.conf
```

`spdadd` tells `setkey(8)` that we want to add a rule to the secure policy database. The rest of this line specifies which packets will match this policy. `A.B.C.D/32` and `W.X.Y.Z/32` are the IP addresses and netmasks that identify the network or hosts that this policy will apply to. In this case, we want it to apply to traffic between these two hosts. `ipencap` tells the kernel that this policy should only apply to packets that encapsulate other packets. `-P out` says that this policy applies to outgoing packets, and `ipsec` says that the packet will be secured.

The second line specifies how this packet will be encrypted. `esp` is the protocol that will be used, while `tunnel` indicates that the packet will be further encapsulated in an IPsec packet. The repeated use of `A.B.C.D` and `W.X.Y.Z`

is used to select the security association to use, and the final `require` mandates that packets must be encrypted if they match this rule.

This rule only matches outgoing packets. You will need a similar rule to match incoming packets.

```
spdadd W.X.Y.Z/32 A.B.C.D/32 ipencap -P in ipsec esp/tunnel/W.X.Y.Z-A.B.C.D/require;
```

Note the `in` instead of `out` in this case, and the necessary reversal of the IP addresses.

The other gateway host (which has the public IP address `W.X.Y.Z`) will need similar rules.

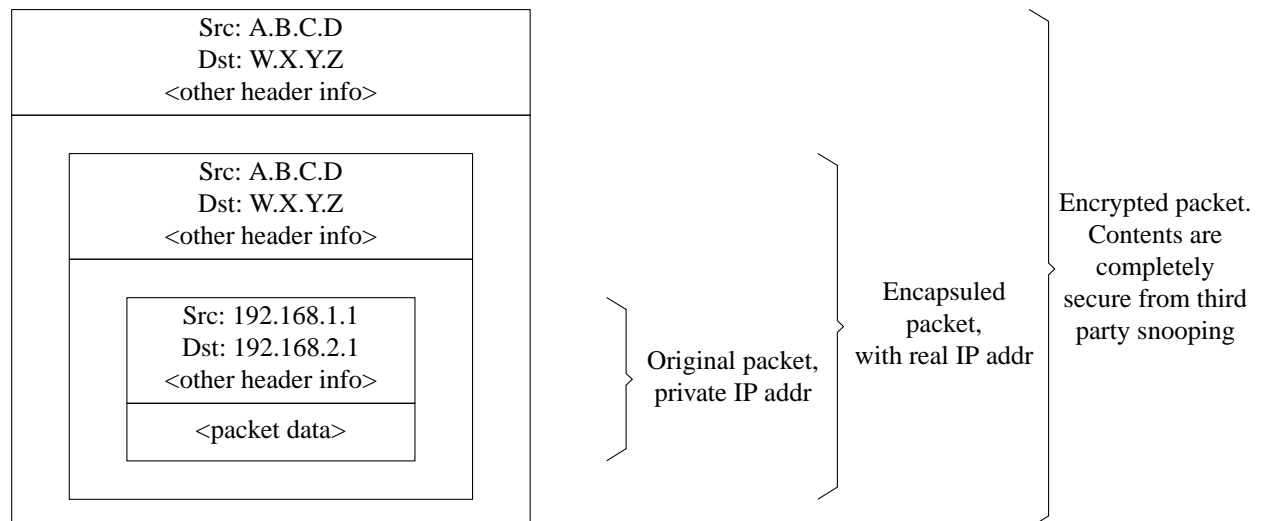
```
spdadd W.X.Y.Z/32 A.B.C.D/32 ipencap -P out ipsec esp/tunnel/W.X.Y.Z-A.B.C.D/require;
spdadd A.B.C.D/32 W.X.Y.Z/32 ipencap -P in ipsec esp/tunnel/A.B.C.D-W.X.Y.Z/require;
```

Finally, you need to add firewall rules to allow ESP and IPENCAP packets back and forth. These rules will need to be added to both hosts.

```
ipfw add 1 allow esp from A.B.C.D to W.X.Y.Z
ipfw add 1 allow esp from W.X.Y.Z to A.B.C.D
ipfw add 1 allow ipencap from A.B.C.D to W.X.Y.Z
ipfw add 1 allow ipencap from W.X.Y.Z to A.B.C.D
```

Because the rules are symmetric you can use the same rules on each gateway host.

Outgoing packets will now look something like this:



When they are received by the far end of the VPN they will first be decrypted (using the security associations that have been negotiated by `racoon`). Then they will enter the `gif` interface, which will unwrap the second layer, until you are left with the innermost packet, which can then travel in to the inner network.

You can check the security using the same `ping(8)` test from earlier. First, log in to the `A.B.C.D` gateway machine, and run:

```
tcpdump dst host 192.168.2.1
```

In another log in session on the same host run

```
ping 192.168.2.1
```

This time you should see output like the following:

```
XXX tcpdump output
```

Now, as you can see, `tcpdump(1)` shows the ESP packets. If you try to examine them with the `-s` option you will see (apparently) gibberish, because of the encryption.

Congratulations. You have just set up a VPN between two remote sites.

Summary

- Configure both kernels with:

```
options IPSEC
options IPSEC_ESP
```

- Install `security/ipsec-tools`. Edit `${PREFIX}/etc/racoon/psk.txt` on both gateway hosts, adding an entry for the remote host's IP address and a secret key that they both know. Make sure this file is mode 0600.
- Add the following lines to `/etc/rc.conf` on each host:

```
ipsec_enable="YES"
ipsec_file="/etc/ipsec.conf"
```

- Create an `/etc/ipsec.conf` on each host that contains the necessary `spdadd` lines. On gateway host #1 this would be:

```
spdadd A.B.C.D/32 W.X.Y.Z/32 ipencap -P out ipsec
      esp/tunnel/A.B.C.D-W.X.Y.Z/require;
spdadd W.X.Y.Z/32 A.B.C.D/32 ipencap -P in ipsec
      esp/tunnel/W.X.Y.Z-A.B.C.D/require;
```

On gateway host #2 this would be:

```
spdadd W.X.Y.Z/32 A.B.C.D/32 ipencap -P out ipsec
      esp/tunnel/W.X.Y.Z-A.B.C.D/require;
spdadd A.B.C.D/32 W.X.Y.Z/32 ipencap -P in ipsec
      esp/tunnel/A.B.C.D-W.X.Y.Z/require;
```

- Add firewall rules to allow IKE, ESP, and IPENCAP traffic to both hosts:

```
ipfw add 1 allow udp from A.B.C.D to W.X.Y.Z isakmp
ipfw add 1 allow udp from W.X.Y.Z to A.B.C.D isakmp
ipfw add 1 allow esp from A.B.C.D to W.X.Y.Z
ipfw add 1 allow esp from W.X.Y.Z to A.B.C.D
ipfw add 1 allow ipencap from A.B.C.D to W.X.Y.Z
ipfw add 1 allow ipencap from W.X.Y.Z to A.B.C.D
```

The previous two steps should suffice to get the VPN up and running. Machines on each network will be able to refer to one another using IP addresses, and all traffic across the link will be automatically and securely encrypted.

14.11. OpenSSH

Contributed by Chern Lee.

OpenSSH is a set of network connectivity tools used to access remote machines securely. It can be used as a direct replacement for `rlogin`, `rsh`, `rcp`, and `telnet`. Additionally, TCP/IP connections can be tunneled/forwarded securely through SSH. **OpenSSH** encrypts all traffic to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks.

OpenSSH is maintained by the OpenBSD project, and is based upon SSH v1.2.12 with all the recent bug fixes and updates. It is compatible with both SSH protocols 1 and 2.

14.11.1. Advantages of Using OpenSSH

Normally, when using `telnet(1)` or `rlogin(1)`, data is sent over the network in a clear, un-encrypted form. Network sniffers anywhere in between the client and server can steal your user/password information or data transferred in your session. **OpenSSH** offers a variety of authentication and encryption methods to prevent this from happening.

14.11.2. Enabling sshd

The **sshd** is an option presented during a Standard install of FreeBSD. To see if **sshd** is enabled, check the `rc.conf` file for:

```
sshd_enable="YES"
```

This will load `sshd(8)`, the daemon program for **OpenSSH**, the next time your system initializes. Alternatively, it is possible to use `/etc/rc.d/sshd rc(8)` script to start **OpenSSH**:

```
/etc/rc.d/sshd start
```

14.11.3. SSH Client

The `ssh(1)` utility works similarly to `rlogin(1)`.

```
# ssh user@example.com
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'example.com' added to the list of known hosts.
user@example.com's password: *****
```

The login will continue just as it would have if a session was created using `rlogin` or `telnet`. SSH utilizes a key fingerprint system for verifying the authenticity of the server when the client connects. The user is prompted to enter `yes` only when connecting for the first time. Future attempts to login are all verified against the saved fingerprint key. The SSH client will alert you if the saved fingerprint differs from the received fingerprint on future login attempts. The fingerprints are saved in `~/.ssh/known_hosts`, or `~/.ssh/known_hosts2` for SSH v2 fingerprints.

By default, recent versions of the **OpenSSH** servers only accept SSH v2 connections. The client will use version 2 if possible and will fall back to version 1. The client can also be forced to use one or the other by passing it the `-1` or `-2` for version 1 or version 2, respectively. The version 1 compatibility is maintained in the client for backwards compatibility with older versions.

14.11.4. Secure Copy

The `scp(1)` command works similarly to `rcp(1)`; it copies a file to or from a remote machine, except in a secure fashion.

```
# scp user@example.com:/COPYRIGHT COPYRIGHT
user@example.com's password: *****
COPYRIGHT          100% | ***** | 4735
00:00
#
```

Since the fingerprint was already saved for this host in the previous example, it is verified when using `scp(1)` here.

The arguments passed to `scp(1)` are similar to `cp(1)`, with the file or files in the first argument, and the destination in the second. Since the file is fetched over the network, through SSH, one or more of the file arguments takes on the form `user@host:<path_to_remote_file>`.

14.11.5. Configuration

The system-wide configuration files for both the **OpenSSH** daemon and client reside within the `/etc/ssh` directory. `ssh_config` configures the client settings, while `sshd_config` configures the daemon.

Additionally, the `sshd_program (/usr/sbin/sshd)` by default, and `sshd_flags rc.conf` options can provide more levels of configuration.

14.11.6. ssh-keygen

Instead of using passwords, `ssh-keygen(1)` can be used to generate DSA or RSA keys to authenticate a user:

```
% ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
bb:48:db:f2:93:57:80:b6:aa:bc:f5:d5:ba:8f:79:17 user@host.example.com
```

`ssh-keygen(1)` will create a public and private key pair for use in authentication. The private key is stored in `~/.ssh/id_dsa` or `~/.ssh/id_rsa`, whereas the public key is stored in `~/.ssh/id_dsa.pub` or `~/.ssh/id_rsa.pub`, respectively for DSA and RSA key types. The public key must be placed in `~/.ssh/authorized_keys` of the remote machine in order for the setup to work. Similarly, RSA version 1 public keys should be placed in `~/.ssh/authorized_keys`.

This will allow connection to the remote machine based upon SSH keys instead of passwords.

If a passphrase is used in `ssh-keygen(1)`, the user will be prompted for a password each time in order to use the private key. `ssh-agent(1)` can alleviate the strain of repeatedly entering long passphrases, and is explored in the Sekcja 14.11.7 section below.

Ostrzeżenie The various options and files can be different according to the **OpenSSH** version you have on your system; to avoid problems you should consult the `ssh-keygen(1)` manual page.

14.11.7. ssh-agent and ssh-add

The `ssh-agent(1)` and `ssh-add(1)` utilities provide methods for **SSH** keys to be loaded into memory for use, without needing to type the passphrase each time.

The `ssh-agent(1)` utility will handle the authentication using the private key(s) that are loaded into it. `ssh-agent(1)` should be used to launch another application. At the most basic level, it could spawn a shell or at a more advanced level, a window manager.

To use `ssh-agent(1)` in a shell, first it will need to be spawned with a shell as an argument. Secondly, the identity needs to be added by running `ssh-add(1)` and providing it the passphrase for the private key. Once these steps have been completed the user will be able to `ssh(1)` to any host that has the corresponding public key installed. For example:

```
% ssh-agent csh
% ssh-add
Enter passphrase for /home/user/.ssh/id_dsa:
Identity added: /home/user/.ssh/id_dsa (/home/user/.ssh/id_dsa)
%
```

To use `ssh-agent(1)` in X11, a call to `ssh-agent(1)` will need to be placed in `~/.xinitrc`. This will provide the `ssh-agent(1)` services to all programs launched in X11. An example `~/.xinitrc` file might look like this:

```
exec ssh-agent startxfce4
```

This would launch `ssh-agent(1)`, which would in turn launch **XFCE**, every time X11 starts. Then once that is done and X11 has been restarted so that the changes can take effect, simply run `ssh-add(1)` to load all of your **SSH** keys.

14.11.8. SSH Tunneling

OpenSSH has the ability to create a tunnel to encapsulate another protocol in an encrypted session.

The following command tells `ssh(1)` to create a tunnel for **telnet**:

```
% ssh -2 -N -f -L 5023:localhost:23 user@foo.example.com
%
```

The `ssh` command is used with the following options:

`-2`

Forces `ssh` to use version 2 of the protocol. (Do not use if you are working with older **SSH** servers)

`-N`

Indicates no command, or tunnel only. If omitted, `ssh` would initiate a normal session.

-f

Forces ssh to run in the background.

-L

Indicates a local tunnel in *localport:remotehost:remoteport* fashion.

user@foo.example.com

The remote SSH server.

An SSH tunnel works by creating a listen socket on localhost on the specified port. It then forwards any connection received on the local host/port via the SSH connection to the specified remote host and port.

In the example, port 5023 on localhost is being forwarded to port 23 on localhost of the remote machine. Since 23 is **telnet**, this would create a secure **telnet** session through an SSH tunnel.

This can be used to wrap any number of insecure TCP protocols such as SMTP, POP3, FTP, etc.

Przykład 14-1. Using SSH to Create a Secure Tunnel for SMTP

```
% ssh -2 -N -f -L 5025:localhost:25 user@mailserver.example.com
user@mailserver.example.com's password: *****
% telnet localhost 5025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mailserver.example.com ESMTP
```

This can be used in conjunction with an ssh-keygen(1) and additional user accounts to create a more seamless/hassle-free SSH tunneling environment. Keys can be used in place of typing a password, and the tunnels can be run as a separate user.

14.11.8.1. Practical SSH Tunneling Examples

14.11.8.1.1. Secure Access of a POP3 Server

At work, there is an SSH server that accepts connections from the outside. On the same office network resides a mail server running a POP3 server. The network, or network path between your home and office may or may not be completely trustable. Because of this, you need to check your e-mail in a secure manner. The solution is to create an SSH connection to your office's SSH server, and tunnel through to the mail server.

```
% ssh -2 -N -f -L 2110:mail.example.com:110 user@ssh-server.example.com
user@ssh-server.example.com's password: *****
```

When the tunnel is up and running, you can point your mail client to send POP3 requests to localhost port 2110. A connection here will be forwarded securely across the tunnel to mail.example.com.

14.11.8.1.2. Bypassing a Draconian Firewall

Some network administrators impose extremely draconian firewall rules, filtering not only incoming connections, but outgoing connections. You may be only given access to contact remote machines on ports 22 and 80 for SSH and web surfing.

You may wish to access another (perhaps non-work related) service, such as an Ogg Vorbis server to stream music. If this Ogg Vorbis server is streaming on some other port than 22 or 80, you will not be able to access it.

The solution is to create an SSH connection to a machine outside of your network's firewall, and use it to tunnel to the Ogg Vorbis server.

```
% ssh -2 -N -f -L 8888:music.example.com:8000 user@unfirewalled-system.example.org
user@unfirewalled-system.example.org's password: *****
```

Your streaming client can now be pointed to localhost port 8888, which will be forwarded over to music.example.com port 8000, successfully evading the firewall.

14.11.9. The `AllowUsers` Users Option

It is often a good idea to limit which users can log in and from where. The `AllowUsers` option is a good way to accomplish this. For example, to only allow the `root` user to log in from `192.168.1.32`, something like this would be appropriate in the `/etc/ssh/sshd_config` file:

```
AllowUsers root@192.168.1.32
```

To allow the user `admin` to log in from anywhere, just list the username by itself:

```
AllowUsers admin
```

Multiple users should be listed on the same line, like so:

```
AllowUsers root@192.168.1.32 admin
```

Notatka: It is important that you list each user that needs to log in to this machine; otherwise they will be locked out.

After making changes to `/etc/ssh/sshd_config` you must tell `sshd(8)` to reload its config files, by running:

```
# /etc/rc.d/sshd reload
```

14.11.10. Further Reading

OpenSSH (<http://www.openssh.com/>)

`ssh(1)` `scp(1)` `ssh-keygen(1)` `ssh-agent(1)` `ssh-add(1)` `ssh_config(5)`

`sshd(8)` `sftp-server(8)` `sshd_config(5)`

14.12. File System Access Control Lists

Contributed by Tom Rhodes.

In conjunction with file system enhancements like snapshots, FreeBSD 5.0 and later offers the security of File System Access Control Lists (ACLs).

Access Control Lists extend the standard UNIX permission model in a highly compatible (POSIX.1e) way. This feature permits an administrator to make use of and take advantage of a more sophisticated security model.

To enable ACL support for UFS file systems, the following:

```
options UFS_ACL
```

must be compiled into the kernel. If this option has not been compiled in, a warning message will be displayed when attempting to mount a file system supporting ACLs. This option is included in the `GENERIC` kernel. ACLs rely on extended attributes being enabled on the file system. Extended attributes are natively supported in the next generation UNIX file system, UFS2.

Notatka: A higher level of administrative overhead is required to configure extended attributes on UFS1 than on UFS2. The performance of extended attributes on UFS2 is also substantially higher. As a result, UFS2 is generally recommended in preference to UFS1 for use with access control lists.

ACLs are enabled by the mount-time administrative flag, `acls`, which may be added to `/etc/fstab`. The mount-time flag can also be automatically set in a persistent manner using `tunefs(8)` to modify a superblock ACLs flag in the file system header. In general, it is preferred to use the superblock flag for several reasons:

- The mount-time ACLs flag cannot be changed by a remount (`mount(8) -u`), only by means of a complete `umount(8)` and fresh `mount(8)`. This means that ACLs cannot be enabled on the root file system after boot. It also means that you cannot change the disposition of a file system once it is in use.
- Setting the superblock flag will cause the file system to always be mounted with ACLs enabled even if there is not an `fstab` entry or if the devices re-order. This prevents accidental mounting of the file system without ACLs enabled, which can result in ACLs being improperly enforced, and hence security problems.

Notatka: We may change the ACLs behavior to allow the flag to be enabled without a complete fresh `mount(8)`, but we consider it desirable to discourage accidental mounting without ACLs enabled, because you can shoot your feet quite nastily if you enable ACLs, then disable them, then re-enable them without flushing the extended attributes. In general, once you have enabled ACLs on a file system, they should not be disabled, as the resulting file protections may not be compatible with those intended by the users of the system, and re-enabling ACLs may re-attach the previous ACLs to files that have since had their permissions changed, resulting in other unpredictable behavior.

File systems with ACLs enabled will show a + (plus) sign in their permission settings when viewed. For example:

```
drwx-----  2 robert  robert  512 Dec 27 11:54 private
drwxrwx---+  2 robert  robert  512 Dec 23 10:57 directory1
drwxrwx---+  2 robert  robert  512 Dec 22 10:20 directory2
drwxrwx---+  2 robert  robert  512 Dec 27 11:57 directory3
drwxr-xr-x   2 robert  robert  512 Nov 10 11:54 public_html
```

Here we see that the `directory1`, `directory2`, and `directory3` directories are all taking advantage of ACLs. The `public_html` directory is not.

14.12.1. Making Use of ACLs

The file system ACLs can be viewed by the `getfacl(1)` utility. For instance, to view the ACL settings on the `test` file, one would use the command:

```
% getfacl test
#file:test
#owner:1001
#group:1001
user::rw-
group::r--
other::r--
```

To change the ACL settings on this file, invoke the `setfacl(1)` utility. Observe:

```
% setfacl -k test
```

The `-k` flag will remove all of the currently defined ACLs from a file or file system. The more preferable method would be to use `-b` as it leaves the basic fields required for ACLs to work.

```
% setfacl -m u:trhodes:rw,group:web:r--,o:--- test
```

In the aforementioned command, the `-m` option was used to modify the default ACL entries. Since there were no pre-defined entries, as they were removed by the previous command, this will restore the default options and assign the options listed. Take care to notice that if you add a user or group which does not exist on the system, an `Invalid argument` error will be printed to `stdout`.

14.13. Monitoring Third Party Security Issues

Contributed by Tom Rhodes.

In recent years, the security world has made many improvements to how vulnerability assessment is handled. The threat of system intrusion increases as third party utilities are installed and configured for virtually any operating system available today.

Vulnerability assessment is a key factor in security, and while FreeBSD releases advisories for the base system, doing so for every third party utility is beyond the FreeBSD Project's capability. There is a way to mitigate third party vulnerabilities and warn administrators of known security issues. A FreeBSD add on utility known as **Portaudit** exists solely for this purpose.

The `security/portaudit` port polls a database, updated and maintained by the FreeBSD Security Team and ports developers, for known security issues.

To begin using **Portaudit**, one must install it from the Ports Collection:

```
# cd /usr/ports/security/portaudit && make install clean
```

During the install process, the configuration files for `periodic(8)` will be updated, permitting **Portaudit** output in the daily security runs. Ensure the daily security run emails, which are sent to `root`'s email account, are being read. No more configuration will be required here.

After installation, an administrator can update the database and view known vulnerabilities in installed packages by invoking the following command:

```
# portaudit -Fda
```

Notatka: The database will automatically be updated during the `periodic(8)` run; thus, the previous command is completely optional. It is only required for the following examples.

To audit the third party utilities installed as part of the Ports Collection at anytime, an administrator need only run the following command:

```
# portaudit -a
```

Portaudit will produce something like this for vulnerable packages:

```
Affected package: cups-base-1.1.22.0_1
Type of problem: cups-base -- HPGL buffer overflow vulnerability.
Reference: <http://www.FreeBSD.org/ports/portaudit/40a3bca2-6809-11d9-a9e7-0001020eed82.html>
```

```
1 problem(s) in your installed packages found.
```

```
You are advised to update or deinstall the affected package(s) immediately.
```

By pointing a web browser to the URL shown, an administrator may obtain more information about the vulnerability in question. This will include versions affected, by FreeBSD Port version, along with other web sites which may contain security advisories.

In short, **Portaudit** is a powerful utility and extremely useful when coupled with the **Portupgrade** port.

14.14. FreeBSD Security Advisories

Contributed by Tom Rhodes.

Like many production quality operating systems, FreeBSD publishes „Security Advisories”. These advisories are usually mailed to the security lists and noted in the Errata only after the appropriate releases have been patched. This section will work to explain what an advisory is, how to understand it, and what measures to take in order to patch a system.

14.14.1. What does an advisory look like?

The FreeBSD security advisories look similar to the one below, taken from the `freebsd-security-notifications` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications>) mailing list.

```
=====
FreeBSD-SA-XX:XX.UTIL                               Security Advisory
```

The FreeBSD Project

```

Topic:          denial of service due to some problem❶

Category:       core❷
Module:         sys❸
Announced:     2003-09-23❹
Credits:        Person@EMAIL-ADDRESS❺
Affects:        All releases of FreeBSD❻
                FreeBSD 4-STABLE prior to the correction date
Corrected:      2003-09-23 16:42:59 UTC (RELENG_4, 4.9-PRERELEASE)
                2003-09-23 20:08:42 UTC (RELENG_5_1, 5.1-RELEASE-p6)
                2003-09-23 20:07:06 UTC (RELENG_5_0, 5.0-RELEASE-p15)
                2003-09-23 16:44:58 UTC (RELENG_4_8, 4.8-RELEASE-p8)
                2003-09-23 16:47:34 UTC (RELENG_4_7, 4.7-RELEASE-p18)
                2003-09-23 16:49:46 UTC (RELENG_4_6, 4.6-RELEASE-p21)
                2003-09-23 16:51:24 UTC (RELENG_4_5, 4.5-RELEASE-p33)
                2003-09-23 16:52:45 UTC (RELENG_4_4, 4.4-RELEASE-p43)
                2003-09-23 16:54:39 UTC (RELENG_4_3, 4.3-RELEASE-p39)❼
CVE Name:       CVE-XXXX-XXXX❽

```

For general information regarding FreeBSD Security Advisories, including descriptions of the fields above, security branches, and the following sections, please visit <http://www.FreeBSD.org/security/>.

I. Background❾

II. Problem Description(10)

III. Impact(11)

IV. Workaround(12)

V. Solution(13)

VI. Correction details(14)

VII. References(15)

❶ The Topic field indicates exactly what the problem is. It is basically an introduction to the current security advisory and notes the utility with the vulnerability.

❷ The Category refers to the affected part of the system which may be one of core, contrib, or ports. The core category means that the vulnerability affects a core component of the FreeBSD operating system. The contrib category means that the vulnerability affects software contributed to the FreeBSD Project, such as

sendmail. Finally the `ports` category indicates that the vulnerability affects add on software available as part of the Ports Collection.

- ③ The `Module` field refers to the component location, for instance `sys`. In this example, we see that the module, `sys`, is affected; therefore, this vulnerability affects a component used within the kernel.
- ④ The `Announced` field reflects the date said security advisory was published, or announced to the world. This means that the security team has verified that the problem does exist and that a patch has been committed to the FreeBSD source code repository.
- ⑤ The `Credits` field gives credit to the individual or organization who noticed the vulnerability and reported it.
- ⑥ The `Affects` field explains which releases of FreeBSD are affected by this vulnerability. For the kernel, a quick look over the output from `ident` on the affected files will help in determining the revision. For ports, the version number is listed after the port name in `/var/db/pkg`. If the system does not sync with the FreeBSD CVS repository and rebuild daily, chances are that it is affected.
- ⑦ The `Corrected` field indicates the date, time, time offset, and release that was corrected.
- ⑧ Reserved for the identification information used to look up vulnerabilities in the Common Vulnerabilities Database system.
- ⑨ The `Background` field gives information on exactly what the affected utility is. Most of the time this is why the utility exists in FreeBSD, what it is used for, and a bit of information on how the utility came to be.
- (10) The `Problem Description` field explains the security hole in depth. This can include information on flawed code, or even how the utility could be maliciously used to open a security hole.
- (11) The `Impact` field describes what type of impact the problem could have on a system. For example, this could be anything from a denial of service attack, to extra privileges available to users, or even giving the attacker superuser access.
- (12) The `Workaround` field offers a feasible workaround to system administrators who may be incapable of upgrading the system. This may be due to time constraints, network availability, or a slew of other reasons. Regardless, security should not be taken lightly, and an affected system should either be patched or the security hole workaround should be implemented.
- (13) The `Solution` field offers instructions on patching the affected system. This is a step by step tested and verified method for getting a system patched and working securely.
- (14) The `Correction Details` field displays the CVS branch or release name with the periods changed to underscore characters. It also shows the revision number of the affected files within each branch.
- (15) The `References` field usually offers sources of other information. This can included web URLs, books, mailing lists, and newsgroups.

14.15. Process Accounting

Contributed by Tom Rhodes.

Process accounting is a security method in which an administrator may keep track of system resources used, their allocation among users, provide for system monitoring, and minimally track a user's commands.

This indeed has its own positive and negative points. One of the positives is that an intrusion may be narrowed down to the point of entry. A negative is the amount of logs generated by process accounting, and the disk space they may require. This section will walk an administrator through the basics of process accounting.

14.15.1. Enable and Utilizing Process Accounting

Before making use of process accounting, it must be enabled. To do this, execute the following commands:

```
# touch /var/account/acct
# accton /var/account/acct
# echo 'accounting_enable="YES"' >> /etc/rc.conf
```

Once enabled, accounting will begin to track CPU stats, commands, etc. All accounting logs are in a non-human readable format and may be viewed using the `sa(8)` utility. If issued without any options, `sa` will print information relating to the number of per user calls, the total elapsed time in minutes, total CPU and user time in minutes, average number of I/O operations, etc.

To view information about commands being issued, one would use the `lastcomm(1)` utility. The `lastcomm` may be used to print out commands issued by users on specific `tty(5)`, for example:

```
# lastcomm ls
trhodes tty1
```

Would print out all known usage of the `ls` by `trhodes` on the `tty1` terminal.

Many other useful options exist and are explained in the `lastcomm(1)`, `acct(5)` and `sa(8)` manual pages.

Przypisy

1. Under FreeBSD the standard login password may be up to 128 characters in length.

Rozdział 15. Mandatory Access Control

Written by Tom Rhodes.

15.1. Synopsis

FreeBSD 5.X introduced new security extensions from the TrustedBSD project based on the POSIX.1e draft. Two of the most significant new security mechanisms are file system Access Control Lists (ACLs) and Mandatory Access Control (MAC) facilities. Mandatory Access Control allows new access control modules to be loaded, implementing new security policies. Some provide protections of a narrow subset of the system, hardening a particular service. Others provide comprehensive labeled security across all subjects and objects. The mandatory part of the definition comes from the fact that the enforcement of the controls is done by administrators and the system, and is not left up to the discretion of users as is done with discretionary access control (DAC, the standard file and System V IPC permissions on FreeBSD).

This chapter will focus on the Mandatory Access Control Framework (MAC Framework), and a set of pluggable security policy modules enabling various security mechanisms.

After reading this chapter, you will know:

- What MAC security policy modules are currently included in FreeBSD and their associated mechanisms.
- What MAC security policy modules implement as well as the difference between a labeled and non-labeled policy.
- How to efficiently configure a system to use the MAC framework.
- How to configure the different security policy modules included with the MAC framework.
- How to implement a more secure environment using the MAC framework and the examples shown.
- How to test the MAC configuration to ensure the framework has been properly implemented.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Rozdział 3).
- Be familiar with the basics of kernel configuration/compilation (Rozdział 8).
- Have some familiarity with security and how it pertains to FreeBSD (Rozdział 14).

Ostrzeżenie The improper use of the information contained herein may cause loss of system access, aggravation of users, or inability to access the features provided by X11. More importantly, MAC should not be relied upon to completely secure a system. The MAC framework only augments existing security policy; without sound security practices and regular security checks, the system will never be completely secure.

It should also be noted that the examples contained within this chapter are just that, examples. It is not recommended that these particular settings be rolled out on a production system. Implementing the various security policy modules takes a good deal of thought and testing. One who does not fully understand exactly how everything works may find him or herself going back through the entire system and reconfiguring many files or directories.

15.1.1. What Will Not Be Covered

This chapter covers a broad range of security issues relating to the MAC framework. The development of new MAC security policy modules will not be covered. A number of security policy modules included with the MAC framework have specific characteristics which are provided for both testing and new module development. These include the `mac_test(4)`, `mac_stub(4)` and `mac_none(4)`. For more information on these security policy modules and the various mechanisms they provide, please review the manual pages.

15.2. Key Terms in this Chapter

Before reading this chapter, a few key terms must be explained. This will hopefully clear up any confusion that may occur and avoid the abrupt introduction of new terms and information.

- *compartment*: A compartment is a set of programs and data to be partitioned or separated, where users are given explicit access to specific components of a system. Also, a compartment represents a grouping, such as a work group, department, project, or topic. Using compartments, it is possible to implement a need-to-know security policy.
- *high water mark*: A high water mark policy is one which permits the raising of security levels for the purpose of accessing higher level information. In most cases, the original level is restored after the process is complete. Currently, the FreeBSD MAC framework does not have a policy for this, but the definition is included for completeness.
- *integrity*: Integrity, as a key concept, is the level of trust which can be placed on data. As the integrity of the data is elevated, so does the ability to trust that data.
- *label*: A label is a security attribute which can be applied to files, directories, or other items in the system. It could be considered a confidentiality stamp; when a label is placed on a file it describes the security properties for that specific file and will only permit access by files, users, resources, etc. with a similar security setting. The meaning and interpretation of label values depends on the policy configuration: while some policies might treat a label as representing the integrity or secrecy of an object, other policies might use labels to hold rules for access.
- *level*: The increased or decreased setting of a security attribute. As the level increases, its security is considered to elevate as well.
- *low water mark*: A low water mark policy is one which permits lowering of the security levels for the purpose of accessing information which is less secure. In most cases, the original security level of the user is restored after the process is complete. The only security policy module in FreeBSD to use this is `mac_lomac(4)`.
- *multilabel*: The `multilabel` property is a file system option which can be set in single user mode using the `tunefs(8)` utility, during the boot operation using the `fstab(5)` file, or during the creation of a new file system. This option will permit an administrator to apply different MAC labels on different objects. This option only applies to security policy modules which support labeling.
- *object*: An object or system object is an entity through which information flows under the direction of a *subject*. This includes directories, files, fields, screens, keyboards, memory, magnetic storage, printers or any other data storage/moving device. Basically, an object is a data container or a system resource; access to an *object* effectively means access to the data.
- *policy*: A collection of rules which defines how objectives are to be achieved. A *policy* usually documents how certain items are to be handled. This chapter will consider the term *policy* in this context as a *security policy*; i.e. a

collection of rules which will control the flow of data and information and define whom will have access to that data and information.

- *sensitivity*: Usually used when discussing MLS. A sensitivity level is a term used to describe how important or secret the data should be. As the sensitivity level increases, so does the importance of the secrecy, or confidentiality of the data.
- *single label*: A single label is when the entire file system uses one label to enforce access control over the flow of data. When a file system has this set, which is any time when the `multilabel` option is not set, all files will conform to the same label setting.
- *subject*: a subject is any active entity that causes information to flow between *objects*; e.g. a user, user processor, system process, etc. On FreeBSD, this is almost always a thread acting in a process on behalf of a user.

15.3. Explanation of MAC

With all of these new terms in mind, consider how the MAC framework augments the security of the system as a whole. The various security policy modules provided by the MAC framework could be used to protect the network and file systems, block users from accessing certain ports and sockets, and more. Perhaps the best use of the policy modules is to blend them together, by loading several security policy modules at a time for a multi-layered security environment. In a multi-layered security environment, multiple policy modules are in effect to keep security in check. This is different to a hardening policy, which typically hardens elements of a system that is used only for specific purposes. The only downside is administrative overhead in cases of multiple file system labels, setting network access control user by user, etc.

These downsides are minimal when compared to the lasting effect of the framework; for instance, the ability to pick and choose which policies are required for a specific configuration keeps performance overhead down. The reduction of support for unneeded policies can increase the overall performance of the system as well as offer flexibility of choice. A good implementation would consider the overall security requirements and effectively implement the various security policy modules offered by the framework.

Thus a system utilizing MAC features should at least guarantee that a user will not be permitted to change security attributes at will; all user utilities, programs and scripts must work within the constraints of the access rules provided by the selected security policy modules; and that total control of the MAC access rules are in the hands of the system administrator.

It is the sole duty of the system administrator to carefully select the correct security policy modules. Some environments may need to limit access control over the network; in these cases, the `mac_portacl(4)`, `mac_ifoff(4)` and even `mac_biba(4)` policy modules might make good starting points. In other cases, strict confidentiality of file system objects might be required. Policy modules such as `mac_bsextended(4)` and `mac_mls(4)` exist for this purpose.

Policy decisions could be made based on network configuration. Perhaps only certain users should be permitted access to facilities provided by `ssh(1)` to access the network or the Internet. The `mac_portacl(4)` would be the policy module of choice for these situations. But what should be done in the case of file systems? Should all access to certain directories be severed from other groups or specific users? Or should we limit user or utility access to specific files by setting certain objects as classified?

In the file system case, access to objects might be considered confidential to some users, but not to others. For an example, a large development team might be broken off into smaller groups of individuals. Developers in project A might not be permitted to access objects written by developers in project B. Yet they might need to access objects created by developers in project C; that is quite a situation indeed. Using the different security policy modules

provided by the MAC framework; users could be divided into these groups and then given access to the appropriate areas without fear of information leakage.

Thus, each security policy module has a unique way of dealing with the overall security of a system. Module selection should be based on a well thought out security policy. In many cases, the overall policy may need to be revised and reimplemented on the system. Understanding the different security policy modules offered by the MAC framework will help administrators choose the best policies for their situations.

The default FreeBSD kernel does not include the option for the MAC framework; thus the following kernel option must be added before trying any of the examples or information in this chapter:

```
options MAC
```

And the kernel will require a rebuild and a reinstall.

Uwaga! While the various manual pages for MAC policy modules state that they may be built into the kernel, it is possible to lock the system out of the network and more. Implementing MAC is much like implementing a firewall, care must be taken to prevent being completely locked out of the system. The ability to revert back to a previous configuration should be considered while the implementation of MAC remotely should be done with extreme caution.

15.4. Understanding MAC Labels

A MAC label is a security attribute which may be applied to subjects and objects throughout the system.

When setting a label, the user must be able to comprehend what it is, exactly, that is being done. The attributes available on an object depend on the policy module loaded, and that policy modules interpret their attributes in different ways. If improperly configured due to lack of comprehension, or the inability to understand the implications, the result will be the unexpected and perhaps, undesired, behavior of the system.

The security label on an object is used as a part of a security access control decision by a policy. With some policies, the label by itself contains all information necessary to make a decision; in other models, the labels may be processed as part of a larger rule set, etc.

For instance, setting the label of `biba/low` on a file will represent a label maintained by the Biba security policy module, with a value of „low”.

A few policy modules which support the labeling feature in FreeBSD offer three specific predefined labels. These are the low, high, and equal labels. Although they enforce access control in a different manner with each policy module, you can be sure that the low label will be the lowest setting, the equal label will set the subject or object to be disabled or unaffected, and the high label will enforce the highest setting available in the Biba and MLS policy modules.

Within single label file system environments, only one label may be used on objects. This will enforce one set of access permissions across the entire system and in many environments may be all that is required. There are a few cases where multiple labels may be set on objects or subjects in the file system. For those cases, the `multilabel` option may be passed to `tunefs(8)`.

In the case of Biba and MLS, a numeric label may be set to indicate the precise level of hierarchical control. This numeric level is used to partition or sort information into different groups of say, classification only permitting access to that group or a higher group level.

In most cases the administrator will only be setting up a single label to use throughout the file system.

Hey wait, this is similar to DAC! I thought MAC gave control strictly to the administrator. That statement still holds true, to some extent as `root` is the one in control and who configures the policies so that users are placed in the appropriate categories/access levels. Alas, many policy modules can restrict the `root` user as well. Basic control over objects will then be released to the group, but `root` may revoke or modify the settings at any time. This is the hierarchal/clearance model covered by policies such as Biba and MLS.

15.4.1. Label Configuration

Virtually all aspects of label policy module configuration will be performed using the base system utilities. These commands provide a simple interface for object or subject configuration or the manipulation and verification of the configuration.

All configuration may be done by use of the `setfmac(8)` and `setpmac(8)` utilities. The `setfmac` command is used to set MAC labels on system objects while the `setpmac` command is used to set the labels on system subjects. Observe:

```
# setfmac biba/high test
```

If no errors occurred with the command above, a prompt will be returned. The only time these commands are not quiescent is when an error occurred; similarly to the `chmod(1)` and `chown(8)` commands. In some cases this error may be a `Permission denied` and is usually obtained when the label is being set or modified on an object which is restricted.¹ The system administrator may use the following commands to overcome this:

```
# setfmac biba/high test
Permission denied
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

As we see above, `setpmac` can be used to override the policy module's settings by assigning a different label to the invoked process. The `getpmac` utility is usually used with currently running processes, such as **sendmail**: although it takes a process ID in place of a command the logic is extremely similar. If users attempt to manipulate a file not in their access, subject to the rules of the loaded policy modules, the `Operation not permitted` error will be displayed by the `mac_set_link` function.

15.4.1.1. Common Label Types

For the `mac_biba(4)`, `mac_mls(4)` and `mac_lomac(4)` policy modules, the ability to assign simple labels is provided. These take the form of `high`, `equal` and `low`, what follows is a brief description of what these labels provide:

- The `low` label is considered the lowest label setting an object or subject may have. Setting this on objects or subjects will block their access to objects or subjects marked `high`.
- The `equal` label should only be placed on objects considered to be exempt from the policy.
- The `high` label grants an object or subject the highest possible setting.

With respect to each policy module, each of those settings will instantiate a different information flow directive. Reading the proper manual pages will further explain the traits of these generic label configurations.

15.4.1.1.1. Advanced Label Configuration

Numeric grade labels are used for `comparison:compartment+compartment`; thus the following:

```
biba/10:2+3+6(5:2+3-20:2+3+4+5+6)
```

May be interpreted as:

„Biba Policy Label”/„Grade 10” : „Compartments 2, 3 and 6”: („grade 5 ...”)

In this example, the first grade would be considered the „effective grade” with „effective compartments”, the second grade is the low grade and the last one is the high grade. In most configurations these settings will not be used; indeed, they offered for more advanced configurations.

When applied to system objects, they will only have a current grade/compartments as opposed to system subjects as they reflect the range of available rights in the system, and network interfaces, where they are used for access control.

The grade and compartments in a subject and object pair are used to construct a relationship referred to as „dominance”, in which a subject dominates an object, the object dominates the subject, neither dominates the other, or both dominate each other. The „both dominate” case occurs when the two labels are equal. Due to the information flow nature of Biba, you have rights to a set of compartments, „need to know”, that might correspond to projects, but objects also have a set of compartments. Users may have to subset their rights using `su` or `setpmac` in order to access objects in a compartment from which they are not restricted.

15.4.1.2. Users and Label Settings

Users themselves are required to have labels so that their files and processes may properly interact with the security policy defined on the system. This is configured through the `login.conf` file by use of login classes. Every policy module that uses labels will implement the user class setting.

An example entry containing every policy module setting is displayed below:

```
default:\
:copyright=/etc/COPYRIGHT:\
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~:/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:\
:manpath=/usr/share/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
```

```
:ignoretime@:\
:label=partition/13,mls/5,biba/10(5-15),lomac/10[2]:
```

The `label` option is used to set the user class default label which will be enforced by MAC. Users will never be permitted to modify this value, thus it can be considered not optional in the user case. In a real configuration, however, the administrator will never wish to enable every policy module. It is recommended that the rest of this chapter be reviewed before any of this configuration is implemented.

Notatka: Users may change their label after the initial login; however, this change is subject constraints of the policy. The example above tells the Biba policy that a process's minimum integrity is 5, its maximum is 15, but the default effective label is 10. The process will run at 10 until it chooses to change label, perhaps due to the user using the `setpmac` command, which will be constrained by Biba to the range set at login.

In all cases, after a change to `login.conf`, the login class capability database must be rebuilt using `cap_mkdb` and this will be reflected throughout every forthcoming example or discussion.

It is useful to note that many sites may have a particularly large number of users requiring several different user classes. In depth planning is required as this may get extremely difficult to manage.

Future versions of FreeBSD will include a new way to deal with mapping users to labels; however, this will not be available until some time after FreeBSD 5.3.

15.4.1.3. Network Interfaces and Label Settings

Labels may also be set on network interfaces to help control the flow of data across the network. In all cases they function in the same way the policies function with respect to objects. Users at high settings in `biba`, for example, will not be permitted to access network interfaces with a label of low.

The `maclabel` may be passed to `ifconfig` when setting the MAC label on network interfaces. For example:

```
# ifconfig bge0 maclabel biba/equal
```

will set the MAC label of `biba/equal` on the `bge(4)` interface. When using a setting similar to `biba/high(low-high)` the entire label should be quoted; otherwise an error will be returned.

Each policy module which supports labeling has a tunable which may be used to disable the MAC label on network interfaces. Setting the label to `equal` will have a similar effect. Review the output from `sysctl`, the policy manual pages, or even the information found later in this chapter for those tunables.

15.4.2. Singlelabel or Multilabel?

By default the system will use the `singlelabel` option. But what does this mean to the administrator? There are several differences which, in their own right, offer pros and cons to the flexibility in the systems security model.

The `singlelabel` only permits for one label, for instance `biba/high` to be used for each subject or object. It provides for lower administration overhead but decreases the flexibility of policies which support labeling. Many administrators may want to use the `multilabel` option in their security policy.

The `multilabel` option will permit each subject or object to have its own independent MAC label in place of the standard `singlelabel` option which will allow only one label throughout the partition. The `multilabel` and

single label options are only required for the policies which implement the labeling feature, including the Biba, Lomac, MLS and SEBSD policies.

In many cases, the `multilabel` may not need to be set at all. Consider the following situation and security model:

- FreeBSD web-server using the MAC framework and a mix of the various policies.
- This machine only requires one label, `biba/high`, for everything in the system. Here the file system would not require the `multilabel` option as a single label will always be in effect.
- But, this machine will be a web server and should have the web server run at `biba/low` to prevent write up capabilities. The Biba policy and how it works will be discussed later, so if the previous comment was difficult to interpret just continue reading and return. The server could use a separate partition set at `biba/low` for most if not all of its runtime state. Much is lacking from this example, for instance the restrictions on data, configuration and user settings; however, this is just a quick example to prove the aforementioned point.

If any of the non-labeling policies are to be used, then the `multilabel` option would never be required. These include the `seeotheruids`, `portacl` and `partition` policies.

It should also be noted that using `multilabel` with a partition and establishing a security model based on `multilabel` functionality could open the doors for higher administrative overhead as everything in the file system would have a label. This includes directories, files, and even device nodes.

The following command will set `multilabel` on the file systems to have multiple labels. This may only be done in single user mode:

```
# tuneefs -l enable /
```

This is not a requirement for the swap file system.

Notatka: Some users have experienced problems with setting the `multilabel` flag on the root partition. If this is the case, please review the Sekcja 15.16 of this chapter.

15.4.3. Controlling MAC with Tunables

Without any modules loaded, there are still some parts of MAC which may be configured using the `sysctl` interface. These tunables are described below and in all cases the number one (1) means enabled while the number zero (0) means disabled:

- `security.mac.enforce_fs` defaults to one (1) and enforces MAC file system policies on the file systems.
- `security.mac.enforce_kld` defaults to one (1) and enforces MAC kernel linking policies on the dynamic kernel linker (see `kld(4)`).
- `security.mac.enforce_network` defaults to one (1) and enforces MAC network policies.
- `security.mac.enforce_pipe` defaults to one (1) and enforces MAC policies on pipes.
- `security.mac.enforce_process` defaults to one (1) and enforces MAC policies on processes which utilize inter-process communication.

- `security.mac.enforce_socket` defaults to one (1) and enforces MAC policies on sockets (see the `socket(2)` manual page).
- `security.mac.enforce_system` defaults to one (1) and enforces MAC policies on system activities such as accounting and rebooting.
- `security.mac.enforce_vm` defaults to one (1) and enforces MAC policies on the virtual memory system.

Notatka: Every policy or MAC option supports tunables. These usually hang off of the `security.mac.<policyname>` tree. To view all of the tunables from MAC use the following command:

```
# sysctl -da | grep mac
```

This should be interpreted as all of the basic MAC policies are enforced by default. If the modules were built into the kernel the system would be extremely locked down and most likely unable to communicate with the local network or connect to the Internet, etc. This is why building the modules into the kernel is not completely recommended. Not because it limits the ability to disable features on the fly with `sysctl`, but it permits the administrator to instantly switch the policies of a system without the requirement of rebuilding and reinstalling a new system.

15.5. Planning the Security Configuration

Whenever a new technology is implemented, a planning phase is always a good idea. During the planning stages, an administrator should in general look at the „big picture”, trying to keep in view at least the following:

- The implementation requirements;
- The implementation goals;

For MAC installations, these include:

- How to classify information and resources available on the target systems.
- What sorts of information or resources to restrict access to along with the type of restrictions that should be applied.
- Which MAC module or modules will be required to achieve this goal.

It is always possible to reconfigure and change the system resources and security settings, it is quite often very inconvenient to search through the system and fix existing files and user accounts. Planning helps to ensure a trouble-free and efficient trusted system implementation. A trial run of the trusted system, including the configuration, is often vital and definitely beneficial *before* a MAC implementation is used on production systems. The idea of just letting loose on a system with MAC is like setting up for failure.

Different environments may have explicit needs and requirements. Establishing an in depth and complete security profile will decrease the need of changes once the system goes live. As such, the future sections will cover the different modules available to administrators; describe their use and configuration; and in some cases provide insight on what situations they would be most suitable for. For instance, a web server might roll out the `mac_biba(4)` and `mac_bsdextended(4)` policies. In other cases, a machine with very few local users, the `mac_partition(4)` might be a good choice.

15.6. Module Configuration

Every module included with the MAC framework may be either compiled into the kernel as noted above or loaded as a run-time kernel module. The recommended method is to add the module name to the `/boot/loader.conf` file so that it will load during the initial boot operation.

The following sections will discuss the various MAC modules and cover their features. Implementing them into a specific environment will also be a consideration of this chapter. Some modules support the use of labeling, which is controlling access by enforcing a label such as „this is allowed and this is not”. A label configuration file may control how files may be accessed, network communication can be exchanged, and more. The previous section showed how the `multilabel` flag could be set on file systems to enable per-file or per-partition access control.

A single label configuration would enforce only one label across the system, that is why the `tunefs` option is called `multilabel`.

15.6.1. The MAC `seeotheruids` Module

Module name: `mac_seeotheruids.ko`

Kernel configuration line: `options MAC_SEEOTHERUIDS`

Boot option: `mac_seeotheruids_load="YES"`

The `mac_seeotheruids(4)` module mimics and extends the `security.bsd.see_other_uids` and `security.bsd.see_other_gids` `sysctl` tunables. This option does not require any labels to be set before configuration and can operate transparently with the other modules.

After loading the module, the following `sysctl` tunables may be used to control the features:

- `security.mac.seeotheruids.enabled` will enable the module's features and use the default settings. These default settings will deny users the ability to view processes and sockets owned by other users.
- `security.mac.seeotheruids.specificgid_enabled` will allow a certain group to be exempt from this policy. To exempt specific groups from this policy, use the `security.mac.seeotheruids.specificgid=xxx` `sysctl` tunable. In the above example, the `xxx` should be replaced with the numeric group ID to be exempted.
- `security.mac.seeotheruids.primarygroup_enabled` is used to exempt specific primary groups from this policy. When using this tunable, the `security.mac.seeotheruids.specificgid_enabled` may not be set.

15.7. The MAC `bsdextended` Module

Module name: `mac_bsdextended.ko`

Kernel configuration line: `options MAC_BSDEXTENDED`

Boot option: `mac_bsdextended_load="YES"`

The `mac_bsdextended(4)` module enforces the file system firewall. This module's policy provides an extension to the standard file system permissions model, permitting an administrator to create a firewall-like ruleset to protect files, utilities, and directories in the file system hierarchy. When access to a file system object is attempted, the list of rules is iterated until either a matching rule is located or the end is reached. This behavior may be changed by the use of a `sysctl(8)` parameter, `security.mac.bsdextended.firstmatch_enabled`. Similar to other firewall modules in FreeBSD, a file containing access control rules can be created and read by the system at boot time using an `rc.conf(5)` variable.

The rule list may be entered using a utility, `ugidfw(8)`, that has a syntax similar to that of `ipfw(8)`. More tools can be written by using the functions in the `libugidfw(3)` library.

Extreme caution should be taken when working with this module; incorrect use could block access to certain parts of the file system.

15.7.1. Examples

After the `mac_bsdextended(4)` module has been loaded, the following command may be used to list the current rule configuration:

```
# ugidfw list
0 slots, 0 rules
```

As expected, there are no rules defined. This means that everything is still completely accessible. To create a rule which will block all access by users but leave `root` unaffected, simply run the following command:

```
# ugidfw add subject not uid root new object not uid root mode n
```

Notatka: In releases prior to FreeBSD 5.3, the `add` parameter did not exist. In those cases the `set` should be used instead. See below for a command example.

This is a very bad idea as it will block all users from issuing even the most simple commands, such as `ls`. A more patriotic list of rules might be:

```
# ugidfw set 2 subject uid user1 object uid user2 mode n
# ugidfw set 3 subject uid user1 object gid user2 mode n
```

This will block any and all access, including directory listings, to `user2`'s home directory from the username `user1`.

In place of `user1`, the `not uid user2` could be passed. This will enforce the same access restrictions above for all users in place of just one user.

Notatka: The `root` user will be unaffected by these changes.

This should provide a general idea of how the `mac_bsdextended(4)` module may be used to help fortify a file system. For more information, see the `mac_bsdextended(4)` and the `ugidfw(8)` manual pages.

15.8. The MAC ifoff Module

Module name: `mac_ifoff.ko`

Kernel configuration line: `options MAC_IFOFF`

Boot option: `mac_ifoff_load="YES"`

The `mac_ifoff(4)` module exists solely to disable network interfaces on the fly and keep network interfaces from being brought up during the initial system boot. It does not require any labels to be set up on the system, nor does it have a dependency on other MAC modules.

Most of the control is done through the `sysctl` tunables listed below.

- `security.mac.ifoff.lo_enabled` will enable/disable all traffic on the loopback (`lo(4)`) interface.
- `security.mac.ifoff.bpfrecv_enabled` will enable/disable all traffic on the Berkeley Packet Filter interface (`bpf(4)`)
- `security.mac.ifoff.other_enabled` will enable/disable traffic on all other interfaces.

One of the most common uses of `mac_ifoff(4)` is network monitoring in an environment where network traffic should not be permitted during the boot sequence. Another suggested use would be to write a script which uses `security/aide` to automatically block network traffic if it finds new or altered files in protected directories.

15.9. The MAC portacl Module

Module name: `mac_portacl.ko`

Kernel configuration line: `MAC_PORTACL`

Boot option: `mac_portacl_load="YES"`

The `mac_portacl(4)` module is used to limit binding to local TCP and UDP ports using a variety of `sysctl` variables. In essence `mac_portacl(4)` makes it possible to allow non-`root` users to bind to specified privileged ports, i.e. ports fewer than 1024.

Once loaded, this module will enable the MAC policy on all sockets. The following tunables are available:

- `security.mac.portacl.enabled` will enable/disable the policy completely.²
- `security.mac.portacl.port_high` will set the highest port number that `mac_portacl(4)` will enable protection for.
- `security.mac.portacl.suser_exempt` will, when set to a non-zero value, exempt the `root` user from this policy.
- `security.mac.portacl.rules` will specify the actual `mac_portacl` policy; see below.

The actual `mac_portacl` policy, as specified in the `security.mac.portacl.rules` `sysctl`, is a text string of the form: `rule[,rule, ...]` with as many rules as needed. Each rule is of the form: `idtype:id:protocol:port`. The `idtype` parameter can be `uid` or `gid` and used to interpret the `id` parameter as either a user id or group id, respectively. The `protocol` parameter is used to determine if the rule should apply to TCP or UDP by setting the parameter to `tcp` or `udp`. The final `port` parameter is the port number to allow the specified user or group to bind to.

Notatka: Since the ruleset is interpreted directly by the kernel only numeric values can be used for the user ID, group ID, and port parameters. I.e. user, group, and port service names cannot be used.

By default, on UNIX-like systems, ports fewer than 1024 can only be used by/bound to privileged processes, i.e. those run as `root`. For `mac_portacl(4)` to allow non-privileged processes to bind to ports below 1024 this standard

UNIX restriction has to be disabled. This can be accomplished by setting the `sysctl(8)` variables `net.inet.ip.portrange.reservedlow` and `net.inet.ip.portrange.reservedhigh` to zero. See the examples below or review the `mac_portacl(4)` manual page for further information.

15.9.1. Examples

The following examples should illuminate the above discussion a little better:

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0 net.inet.ip.portrange.reservedhigh=0
```

First we set `mac_portacl(4)` to cover the standard privileged ports and disable the normal UNIX bind restrictions.

```
# sysctl security.mac.portacl.suser_exempt=1
```

The `root` user should not be crippled by this policy, thus set the `security.mac.portacl.suser_exempt` to a non-zero value. The `mac_portacl(4)` module has now been set up to behave the same way UNIX-like systems behave by default.

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

Allow the user with UID 80 (normally the `www` user) to bind to port 80. This can be used to allow the `www` user to run a web server without ever having `root` privilege.

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

Permit the user with the UID of 1001 to bind to the TCP ports 110 („pop3”) and 995 („pop3s”). This will permit this user to start a server that accepts connections on ports 110 and 995.

15.10. The MAC partition Module

Module name: `mac_partition.ko`

Kernel configuration line: `options MAC_PARTITION`

Boot option: `mac_partition_load="YES"`

The `mac_partition(4)` policy will drop processes into specific „partitions” based on their MAC label. Think of it as a special type of `jail(8)`, though that is hardly a worthy comparison.

This is one module that should be added to the `loader.conf(5)` file so that it loads and enables the policy during the boot process.

Most configuration for this policy is done using the `setpmac(8)` utility which will be explained below. The following `sysctl` tunable is available for this policy:

- `security.mac.partition.enabled` will enable the enforcement of MAC process partitions.

When this policy is enabled, users will only be permitted to see their processes, and any others within their partition, but will not be permitted to work with utilities outside the scope of this partition. For instance, a user in the

`insecure` class above will not be permitted to access the `top` command as well as many other commands that must spawn a process.

To set or drop utilities into a partition label, use the `setpmac` utility:

```
# setpmac partition/13 top
```

This will add the `top` command to the label set on users in the `insecure` class. Note that all processes spawned by users in the `insecure` class will stay in the `partition/13` label.

15.10.1. Examples

The following command will show you the partition label and the process list:

```
# ps Zax
```

This next command will allow the viewing of another user's process partition label and that user's currently running processes:

```
# ps -ZU trhodes
```

Notatka: Users can see processes in `root`'s label unless the `mac_seeotheruids(4)` policy is loaded.

A really crafty implementation could have all of the services disabled in `/etc/rc.conf` and started by a script that starts them with the proper labeling set.

Notatka: The following policies support integer settings in place of the three default labels offered. These options, including their limitations, are further explained in the module manual pages.

15.11. The MAC Multi-Level Security Module

Module name: `mac_mls.ko`

Kernel configuration line: `options MAC_MLS`

Boot option: `mac_mls_load="YES"`

The `mac_mls(4)` policy controls access between subjects and objects in the system by enforcing a strict information flow policy.

In MLS environments, a „clearance” level is set in each subject or objects label, along with compartments. Since these clearance or sensibility levels can reach numbers greater than six thousand; it would be a daunting task for any system administrator to thoroughly configure each subject or object. Thankfully, three „instant” labels are already included in this policy.

These labels are `mls/low`, `mls/equal` and `mls/high`. Since these labels are described in depth in the manual page, they will only get a brief description here:

- The `mls/low` label contains a low configuration which permits it to be dominated by all other objects. Anything labeled with `mls/low` will have a low clearance level and not be permitted to access information of a higher level. In addition, this label will prevent objects of a higher clearance level from writing or passing information on to them.
- The `mls/equal` label should be placed on objects considered to be exempt from the policy.
- The `mls/high` label is the highest level of clearance possible. Objects assigned this label will hold dominance over all other objects in the system; however, they will not permit the leaking of information to objects of a lower class.

MLS provides for:

- A hierarchical security level with a set of non hierarchical categories;
- Fixed rules: no read up, no write down (a subject can have read access to objects on its own level or below, but not above. Similarly, a subject can have write access to objects on its own level or above but not beneath.);
- Secrecy (preventing inappropriate disclosure of data);
- Basis for the design of systems that concurrently handle data at multiple sensitivity levels (without leaking information between secret and confidential).

The following `sysctl` tunables are available for the configuration of special services and interfaces:

- `security.mac.mls.enabled` is used to enable/disable the MLS policy.
- `security.mac.mls.ptys_equal` will label all `pty(4)` devices as `mls/equal` during creation.
- `security.mac.mls.revocation_enabled` is used to revoke access to objects after their label changes to a label of a lower grade.
- `security.mac.mls.max_compartments` is used to set the maximum number of compartment levels with objects; basically the maximum compartment number allowed on a system.

To manipulate the MLS labels, the `setfmac(8)` command has been provided. To assign a label to an object, issue the following command:

```
# setfmac mls/5 test
```

To get the MLS label for the file `test` issue the following command:

```
# getfmac test
```

This is a summary of the MLS policy's features. Another approach is to create a master policy file in `/etc` which specifies the MLS policy information and to feed that file into the `setfmac` command. This method will be explained after all policies are covered.

15.11.1. Planning Mandatory Sensitivity

With the Multi-Level Security Policy Module, an administrator plans for controlling the flow of sensitive information. By default, with its block read up block write down nature, the system defaults everything to a low state. Everything is accessible and an administrator slowly changes this during the configuration stage; augmenting the confidentiality of the information.

Beyond the three basic label options above, an administrator may group users and groups as required to block the information flow between them. It might be easier to look at the information in clearance levels familiarized with words, for instance classifications such as `Confidential`, `Secret`, and `Top Secret`. Some administrators might just create different groups based on project levels. Regardless of classification method, a well thought out plan must exist before implementing such a restrictive policy.

Some example situations for this security policy module could be an e-commerce web server, a file server holding critical company information, and financial institution environments. The most unlikely place would be a personal workstation with only two or three users.

15.12. The MAC Biba Module

Module name: `mac_biba.ko`

Kernel configuration line: `options MAC_BIBA`

Boot option: `mac_biba_load="YES"`

The `mac_biba(4)` module loads the MAC Biba policy. This policy works much like that of the MLS policy with the exception that the rules for information flow are slightly reversed. This is said to prevent the downward flow of sensitive information whereas the MLS policy prevents the upward flow of sensitive information; thus, much of this section can apply to both policies.

In Biba environments, an „integrity” label is set on each subject or object. These labels are made up of hierarchal grades, and non-hierarchal components. As an object’s or subject’s grade ascends, so does its integrity.

Supported labels are `biba/low`, `biba/equal`, and `biba/high`; as explained below:

- The `biba/low` label is considered the lowest integrity an object or subject may have. Setting this on objects or subjects will block their write access to objects or subjects marked high. They still have read access though.
- The `biba/equal` label should only be placed on objects considered to be exempt from the policy.
- The `biba/high` label will permit writing to objects set at a lower label, but not permit reading that object. It is recommended that this label be placed on objects that affect the integrity of the entire system.

Biba provides for:

- Hierarchical integrity level with a set of non hierarchical integrity categories;
- Fixed rules: no write up, no read down (opposite of MLS). A subject can have write access to objects on its own level or below, but not above. Similarly, a subject can have read access to objects on its own level or above, but not below;
- Integrity (preventing inappropriate modification of data);
- Integrity levels (instead of MLS sensitivity levels).

The following `sysctl` tunables can be used to manipulate the Biba policy.

- `security.mac.biba.enabled` may be used to enable/disable enforcement of the Biba policy on the target machine.
- `security.mac.biba.ptys_equal` may be used to disable the Biba policy on `pty(4)` devices.

- `security.mac.biba.revocation_enabled` will force the revocation of access to objects if the label is changed to dominate the subject.

To access the Biba policy setting on system objects, use the `setfmac` and `getfmac` commands:

```
# setfmac biba/low test
# getfmac test
test: biba/low
```

15.12.1. Planning Mandatory Integrity

Integrity, different from sensitivity, guarantees that the information will never be manipulated by untrusted parties. This includes information passed between subjects, objects, and both. It ensures that users will only be able to modify and in some cases even access information they explicitly need to.

The `mac_biba(4)` security policy module permits an administrator to address which files and programs a user or users may see and invoke while assuring that the programs and files are free from threats and trusted by the system for that user, or group of users.

During the initial planning phase, an administrator must be prepared to partition users into grades, levels, and areas. Users will be blocked access not only to data but programs and utilities both before and after they start. The system will default to a high label once this policy module is enabled, and it is up to the administrator to configure the different grades and levels for users. Instead of using clearance levels as described above, a good planning method could include topics. For instance, only allow developers modification access to the source code repository, source code compiler, and other development utilities. While other users would be grouped into other categories such as testers, designers, or just ordinary users and would only be permitted read access.

With its natural security control, a lower integrity subject is unable to write to a higher integrity subject; a higher integrity subject cannot observe or read a lower integrity object. Setting a label at the lowest possible grade could make it inaccessible to subjects. Some prospective environments for this security policy module would include a constrained web server, development and test machine, and source code repository. A less useful implementation would be a personal workstation, a machine used as a router, or a network firewall.

15.13. The MAC LOMAC Module

Module name: `mac_lomac.ko`

Kernel configuration line: `options MAC_LOMAC`

Boot option: `mac_lomac_load="YES"`

Unlike the MAC Biba policy, the `mac_lomac(4)` policy permits access to lower integrity objects only after decreasing the integrity level to not disrupt any integrity rules.

The MAC version of the Low-watermark integrity policy, not to be confused with the older `lomac(4)` implementation, works almost identically to Biba, but with the exception of using floating labels to support subject demotion via an auxiliary grade compartment. This secondary compartment takes the form of `[auxgrade]`. When assigning a lomac policy with an auxiliary grade, it should look a little bit like: `lomac/10[2]` where the number two (2) is the auxiliary grade.

The MAC LOMAC policy relies on the ubiquitous labeling of all system objects with integrity labels, permitting subjects to read from low integrity objects and then downgrading the label on the subject to prevent future writes to

high integrity objects. This is the `[auxgrade]` option discussed above, thus the policy may provide for greater compatibility and require less initial configuration than Biba.

15.13.1. Examples

Like the Biba and MLS policies; the `setfmac` and `setpmac` utilities may be used to place labels on system objects:

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes lomac/high[low]
```

Notice the auxiliary grade here is `low`, this is a feature provided only by the MAC LOMAC policy.

15.14. Nagios in a MAC Jail

The following demonstration will implement a secure environment using various MAC modules with properly configured policies. This is only a test and should not be considered the complete answer to everyone's security woes. Just implementing a policy and ignoring it never works and could be disastrous in a production environment.

Before beginning this process, the `multilabel` option must be set on each file system as stated at the beginning of this chapter. Not doing so will result in errors. While at it, ensure that the `net-mngt/nagios-plugins`, `net-mngt/nagios`, and `www/apache13` ports are all installed, configured, and working correctly.

15.14.1. Create an insecure User Class

Begin the procedure by adding the following user class to the `/etc/login.conf` file:

```
insecure:\
:copyright=/etc/COPYRIGHT:\
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~:/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
:manpath=/usr/share/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=biba/10(10-10):
```

And adding the following line to the default user class:

```
:label=biba/high:
```

Once this is completed, the following command must be issued to rebuild the database:

```
# cap_mkdb /etc/login.conf
```

15.14.2. Boot Configuration

Do not reboot yet, just add the following lines to `/boot/loader.conf` so the required modules will load during system initialization:

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
```

15.14.3. Configure Users

Set the `root` user to the default class using:

```
# pw usermod root -L default
```

All user accounts that are not `root` or system users will now require a login class. The login class is required otherwise users will be refused access to common commands such as `vi(1)`. The following `sh` script should do the trick:

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' \
    /etc/passwd`; do pw usermod $x -L default; done;
```

Drop the `nagios` and `www` users into the insecure class:

```
# pw usermod nagios -L insecure
# pw usermod www -L insecure
```

15.14.4. Create the Contexts File

A contexts file should now be created; the following example file should be placed in `/etc/policy.contexts`.

```
# This is the default BIBA policy for this system.

# System:
/var/run                biba/equal
/var/run/*              biba/equal

/dev                    biba/equal
/dev/*                  biba/equal

/var                    biba/equal
```

```

/var/spool                biba/equal
/var/spool/*              biba/equal

/var/log                  biba/equal
/var/log/*                biba/equal

/tmp                      biba/equal
/tmp/*                    biba/equal
/var/tmp                  biba/equal
/var/tmp/*                biba/equal

/var/spool/mqueue         biba/equal
/var/spool/clientmqueue   biba/equal

# For Nagios:
/usr/local/etc/nagios
/usr/local/etc/nagios/*   biba/10

/var/spool/nagios         biba/10
/var/spool/nagios/*       biba/10

# For apache
/usr/local/etc/apache     biba/10
/usr/local/etc/apache/*   biba/10

```

This policy will enforce security by setting restrictions on the flow of information. In this specific configuration, users, root and others, should never be allowed to access **Nagios**. Configuration files and processes that are a part of **Nagios** will be completely self contained or jailed.

This file may now be read into our system by issuing the following command:

```

# setfsmac -ef /etc/policy.contexts /
# setfsmac -ef /etc/policy.contexts /

```

Notatka: The above file system layout may be different depending on environment; however, it must be run on every single file system.

The `/etc/mac.conf` file requires the following modifications in the main section:

```

default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba

```

15.14.5. Enable Networking

Add the following line to `/boot/loader.conf`:

```
security.mac.biba.trust_all_interfaces=1
```

And the following to the network card configuration stored in `rc.conf`. If the primary Internet configuration is done via DHCP, this may need to be configured manually after every system boot:

```
maclabel biba/equal
```

15.14.6. Testing the Configuration

Ensure that the web server and **Nagios** will not be started on system initialization, and reboot. Ensure the `root` user cannot access any of the files in the **Nagios** configuration directory. If `root` can issue an `ls(1)` command on `/var/spool/nagios`, then something is wrong. Otherwise a „permission denied” error should be returned.

If all seems well, **Nagios**, **Apache**, and **Sendmail** can now be started in a way fitting of the security policy. The following commands will make this happen:

```
# cd /etc/mail && make stop && \
setpmac biba/equal make start && setpmac biba/10\10-10\ apachectl start && \
setpmac biba/10\10-10\ /usr/local/etc/rc.d/nagios.sh forstart
```

Double check to ensure that everything is working properly. If not, check the log files or error messages. Use the `sysctl(8)` utility to disable the `mac_biba(4)` security policy module enforcement and try starting everything again, like normal.

Notatka: The `root` user can change the security enforcement and edit the configuration files without fear. The following command will permit the degradation of the security policy to a lower grade for a newly spawned shell:

```
# setpmac biba/10 csh
```

To block this from happening, force the user into a range via `login.conf(5)`. If `setpmac(8)` attempts to run a command outside of the compartment's range, an error will be returned and the command will not be executed. In this case, setting `root` to `biba/high(high-high)`.

15.15. User Lock Down

This example considers a relatively small, fewer than fifty users, storage system. Users would have login capabilities, and be permitted to not only store data but access resources as well.

For this scenario, the `mac_bsdextended(4)` mixed with `mac_seeotheruids(4)` could co-exist and block access not only to system objects but to hide user processes as well.

Begin by adding the following lines to `/boot/loader.conf`:

```
mac_seeotheruids_enabled="YES"
```

The `mac_bsdextended(4)` security policy module may be activated through the use of the following `rc.conf` variable:

```
ugidfw_enable="YES"
```

Default rules stored in `/etc/rc.bsdextended` will be loaded at system initialization; however, the default entries may need modification. Since this machine is expected only to service users, everything may be left commented out except the last two. These will force the loading of user owned system objects by default.

Add the required users to this machine and reboot. For testing purposes, try logging in as a different user across two consoles. Run the `ps aux` command to see if processes of other users are visible. Try to run `ls(1)` on another users home directory, it should fail.

Do not try to test with the `root` user unless the specific `sysctls` have been modified to block super user access.

Notatka: When a new user is added, their `mac_bsdextended(4)` rule will not be in the ruleset list. To update the ruleset quickly, simply unload the security policy module and reload it again using the `kldunload(8)` and `kldload(8)` utilities.

15.16. Troubleshooting the MAC Framework

During the development stage, a few users reported problems with normal configuration. Some of these problems are listed below:

15.16.1. The `multilabel` option cannot be enabled on `/`

The `multilabel` flag does not stay enabled on my root (`/`) partition!

It seems that one out of every fifty users has this problem, indeed, we had this problem during our initial configuration. Further observation of this so called „bug” has lead me to believe that it is a result of either incorrect documentation or misinterpretation of the documentation. Regardless of why it happened, the following steps may be taken to resolve it:

1. Edit `/etc/fstab` and set the root partition at `ro` for read-only.
2. Reboot into single user mode.
3. Run `tunefs -l enable on /`.
4. Reboot the system into normal mode.
5. Run `mount -urw /` and change the `ro` back to `rw` in `/etc/fstab` and reboot the system again.
6. Double-check the output from the `mount` to ensure that `multilabel` has been properly set on the root file system.

15.16.2. Cannot start a X11 server after MAC

After establishing a secure environment with MAC, I am no longer able to start X!

This could be caused by the MAC `partition` policy or by a mislabeling in one of the MAC labeling policies. To debug, try the following:

1. Check the error message; if the user is in the `insecure` class, the `partition` policy may be the culprit. Try setting the user's class back to the `default` class and rebuild the database with the `cap_mkdb` command. If this does not alleviate the problem, go to step two.
2. Double-check the label policies. Ensure that the policies are set correctly for the user in question, the X11 application, and the `/dev` entries.
3. If neither of these resolve the problem, send the error message and a description of your environment to the TrustedBSD discussion lists located at the TrustedBSD (<http://www.TrustedBSD.org>) website or to the Ogólna lista dyskusyjna FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) mailing list.

15.16.3. Error: `_secure_path(3)` cannot stat `.login_conf`

When I attempt to switch from the `root` to another user in the system, the error message `_secure_path: unable to state .login_conf`.

This message is usually shown when the user has a higher label setting then that of the user whom they are attempting to become. For instance a user on the system, `joe`, has a default label of `biba/low`. The `root` user, who has a label of `biba/high`, cannot view `joe`'s home directory. This will happen regardless if `root` has used the `su` command to become `joe`, or not. In this scenario, the Biba integrity model will not permit `root` to view objects set at a lower integrity level.

15.16.4. The `root` username is broken!

In normal or even single user mode, the `root` is not recognized. The `whoami` command returns 0 (zero) and `su` returns `who are you?`. What could be going on?

This can happen if a labeling policy has been disabled, either by a `sysctl(8)` or the policy module was unloaded. If the policy is being disabled or has been temporarily disabled, then the login capabilities database needs to be reconfigured with the `label` option being removed. Double check the `login.conf` file to ensure that all `label` options have been removed and rebuild the database with the `cap_mkdb` command.

This may also happen if a policy restricts access to the `master.passwd` file or database. Usually caused by an administrator altering the file under a label which conflicts with the general policy being used by the system. In these cases, the user information would be read by the system and access would be blocked as the file has inherited the new label. Disable the policy via a `sysctl(8)` and everything should return to normal.

Przypisy

1. Other conditions may produce different failures. For instance, the file may not be owned by the user attempting to relabel the object, the object may not exist or may be read only. A mandatory policy will not allow the process to relabel the file, maybe because of a property of the file, a property of the process, or a property of the proposed new label value. For example: a user running at low integrity tries to change the label of a high integrity file. Or perhaps a user running at low integrity tries to change the label of a low integrity file to a high integrity label.
2. Due to a bug the `security.mac.portacl.enabled` `sysctl` variable will not work on FreeBSD 5.2.1 or previous releases.

Rozdział 16. Security Event Auditing

Written by Tom Rhodes.

16.1. Synopsis

The FreeBSD 7-CURRENT development branch includes support for Event Auditing based on the POSIX.1e draft and Sun's published BSM API and file format. Event auditing permits the selective logging of security-relevant system events for the purposes of post-mortem analysis, system monitoring, and intrusion detection. After some settling time in FreeBSD 7-CURRENT, this support will be merged to FreeBSD 6-STABLE and appear in subsequent releases.

Ostrzeżenie The audit facility in FreeBSD is considered experimental, and production deployment should occur only after careful consideration of the risks of deploying experimental software.

This chapter will focus mainly on the installation and configuration of Event Auditing. Explanation of audit policies, and an example configuration will be provided for the convenience of the reader.

After reading this chapter, you will know:

- What Event Auditing is and how it works.
- How to configure Event Auditing on FreeBSD for users and processes.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Rozdział 3).
- Be familiar with the basics of kernel configuration/compilation (Rozdział 8).
- Have some familiarity with security and how it pertains to FreeBSD (Rozdział 14).

Ostrzeżenie Event auditing can generate a great deal of log file data, exceeding gigabytes a week in some configurations. An administrator should read this chapter in its entirety to avoid possible self-inflicted DoS attacks due to improper configuration.

The implementation of Event Auditing in FreeBSD is similar to that of the Sun™ Basic Security Module, or BSM library. Thus, the configuration is almost completely interchangeable with Solaris and Mac OS X/Darwin operating systems.

16.2. Key Terms - Words to Know

Before reading this chapter, a few key terms must be explained. This is intended to clear up any confusion that may occur and to avoid the abrupt introduction of new terms and information.

- *event*: An auditable event is an event that can be logged using the audit subsystem. The administrator can configure which events will be audited. Examples of security-relevant events include the creation of a file, the building of a network connection, or the logging in of a user. Events are either „attributable”, meaning that they can be traced back to a user authentication, or „non-attributable”. Examples of non-attributable events are any events that occur before authentication has succeeded in the login process, such as failed authentication attempts.
- *class*: Events may be assigned to one or more classes, usually based on the general category of the events, such as „file creation”, „file access”, or „network”. Login and logout events are assigned to the `login` class. The use of classes allows the administrator to specify high level auditing rules without having to specify whether each individual auditable operation will be logged.
- *record*: A record is a log entry describing a security event. Records typically have a record event type, information on the subject (user) associated with the event, time information, information on any objects, such as files, and information on whether the event corresponded to a successful operation.
- *trail*: An audit trail, or log file, consists of a series of audit records describing security events. Typically, trails are in roughly chronological order with respect to the time events completed. Only authorized processes are allowed to commit records to the audit trail.
- *prefix*: A prefix is considered to be the configuration element used to toggle auditing for success and failed events.

16.3. Installing Audit Support

Support for Event Auditing is installed with the normal `installworld` process. An administrator may confirm this by viewing the contents of `/etc/security`. Files beginning with the word *audit* should be present. For example, `audit_event`.

In-kernel support for the framework must also exist. This may be done by adding the following lines to the local kernel configuration file:

```
options AUDIT
```

Rebuild and reinstall the kernel via the normal process explained in Rozdział 8.

Once completed, enable the audit daemon by adding the following line to `rc.conf(5)`:

```
auditd_enable="YES"
```

Functionality not provided by the default may be added here with the `auditd_flags` option.

16.4. Audit Configuration

All configuration files for security audit are found in `/etc/security`. The following files must be present before the audit daemon is started:

- `audit_class` - Contains the definitions of the audit classes.
- `audit_control` - Controls aspects of the audit subsystem, such as default audit classes, minimum disk space to leave on the audit log volume, etc.
- `audit_event` - Defines the kernel audit events. These map, mostly, to system calls.

- `audit_user` - The events to audit for individual users. Users not appearing here will be subject to the default configuration in the control configuration file.
- `audit_warn` - A shell script used by `auditd` to generate warning messages in exceptional situations, such as when space for audit records is running low.

16.4.1. Audit File Syntax

The configuration file syntax is rather arcane, albeit easy to work with. One thing an administrator must be leery about is overriding system defaults. This could create potential openings for audit data to not be collected properly.

The audit subsystem will accept both the short name and long name with regards to configuration syntax. A syntax map has been included below.

The following list contains all supported audit classes:

- `all` - `all` - All audit flags set.
- `ad` - `administrative` - Administrative actions performed on the system as a whole.
- `ap` - `application` - Application defined action.
- `cl` - `file_close` - Audit calls to the `close` system call.
- `ex` - `exec` - Audit program or utility execution.
- `fa` - `file_attr_acc` - Audit the access of object attributes such as `stat(1)`, `pathconf(2)` and similar events.
- `fc` - `file_creation` - Audit events where a file is created as a result.
- `fd` - `file_deletion` - Audit events where file deletion occurs.
- `fm` - `file_attr_mod` - Audit events where file attribute modification occurs, such as `chown(8)`, `chflags(1)`, `flock(2)`, etc.
- `fr` - `file_read` - Audit events in which data is read, files are opened for reading, etc.
- `fw` - `file_write` - Audit events in which data is written, files are written or modified, etc.
- `io` - `ioctl` - Audit use of the `ioctl(2)` system call.
- `ip` - `ipc` - Audit various forms of Inter-Process Communication, including POSIX pipes and System V IPC operations.
- `lo` - `login_logout` - Audit `login(1)` and `logout(1)` events occurring on the system.
- `na` - `non_attrib` - Audit non-attributable events.
- `no` - `no_class` - Null class used to disable event auditing.
- `nt` - `network` - Audit events related to network actions, such as `connect(2)` and `accept(2)`.
- `ot` - `other` - Audit miscellaneous events.
- `pc` - `process` - Audit process operations, such as `exec(3)` and `exit(3)`.

Following is a list of all supported audit prefixes:

- `none` - Audit both the success or failure of an event. For example, just listing a class will result in the auditing of both success and failure.

- + - Audit successful events only.
- - - Audit failed events only.

Ostrzeżenie Using the `all` class with either the positive or negative prefix can generate a large amount of data at an extremely rapid rate.

Extra prefixes used to modify the default configuration values:

- ^- - Disable auditing of failed events.
- ^+ - Enable auditing of successful events.
- ^ - Disable auditing of both successful and failed events.

16.4.2. Configuration Files

In most cases, administrators will need to modify only two files when configuring the audit system: `audit_control` and `audit_user`. The first controls system-wide audit parameters and defaults for both attributable and non-attributable events. The second may be used to tune the level and nature of auditing for individual users.

16.4.2.1. The `audit_control` File

The `audit_control` file contains some basic defaults that the administrator may wish to modify. Perhaps even set some new ones. Viewing the contents of this file, we see the following:

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
```

The `dir` option is used to set the default directory where audit logs are stored. Audit is frequently configured so that audit logs are stored on a dedicated file system, so as to prevent interference between the audit subsystem and other subsystems when file systems become full.

The `flags` option is used to set the system-wide defaults. The current setting, `lo` configures the auditing of all `login(1)` and `logout(1)` actions. A more complex example, `lo,ad,-all,^-fa,^-fc,^-cl` audits all system `login(1)` and `logout(1)` actions, all administrator actions, all failed events in the system, and finally disables auditing of failed attempts for `fa`, `fc`, and `cl`. Even though the `-all` turned on the auditing of all failed attempts, the `^-` prefix will override that for the latter options.

Notice that the previous paragraph shows the file is read from left to right. As such, values further on the right side may override a previous value specified to its left.

The `minfree` option defines the minimum percentage of free space for audit file systems. This relates to the file system where audit logs are stored. For example, if the `dir` specifies `/var/audit` and `minfree` is set to twenty (20), warning messages will be generated when the `/var` file system grows to eighty (80) percent full.

The `naflags` option specifies audit classes to be audited for non-attributed events — that is, events for which there is no authenticated user.

16.4.2.2. The `audit_user` File

The `audit_user` file permits the administrator to determine which classes of audit events should be logged for which system users.

The following is the defaults currently placed in the `audit_user` file:

```
root:lo:no
audit:fc:no
```

Notice how the default is to audit all cases of `login/logout` and disable auditing of all other actions for `root`. This configuration also audits all file creation and disables all other auditing for the `audit` user. While event auditing does not require a special user exist, some configurations, specifically environments making use of MAC, may require it.

16.5. Event Audit Administration

Events written by the kernel audit subsystem cannot be altered or read in plain text. Data is stored and accessed in a method similar to that of `ktrace(1)` and `kdump(1)`, that is, they may only be viewed by dumping them using the `praudit` command; audit trails may be reduced using the `auditreduce` command, which selects records from an audit trail based on properties of interest, such as the user, time of the event, and type of operation.

For example, the `praudit` utility will dump the entire contents of a specified audit log in plain text. To dump an audit log in its entirety, use:

```
# praudit /var/audit/AUDITFILE
```

Where `AUDITFILE` is the audit log of viewing choice. Since audit logs may contain enormous amounts of data, an administrator may prefer to select records for specific users. This is made possible with the following command, where `trhodes` is the user of choice:

```
# auditreduce -e trhodes /var/audit/AUDITFILE | praudit
```

This will select all audit records produced by the user `trhodes` stored in the `AUDITFILE` file.

There are several other options available for reading audit records, see the aforementioned command's manual pages for a more in depth explanation.

16.5.1. Rotating Audit Log Files

Due to log reliability requirements, audit trails are written to only by the kernel, and managed only by `auditd`. Administrators should not attempt to use `newsyslog.conf(5)` or other tools to directly rotate audit logs. Instead, the `audit` management tool should be used to shut down auditing, reconfigure the audit system, and perform log rotation. The following command causes the audit daemon to create a new audit log and signal the kernel to switch to using the new log. The old log will be terminated and renamed, at which point it may then be manipulated by the administrator.

```
# audit -n
```

Ostrzeżenie If the `auditd` daemon is not currently running, the previous command will fail and an error message will be produced.

Adding the following line to `/etc/crontab` will force the rotation every twelve hours from `cron(8)`:

```
* * /12 * * * root /usr/sbin/audit -n
```

The change will take effect once you have saved the new `/etc/crontab`.

16.5.2. Delegating Audit Review Rights

By default, only the root user has the right to read system audit logs. However, that right may be delegated to members of the `audit` group, as the audit directory and audit trail files are assigned to that group, and made group-readable. As the ability to track audit log contents provides significant insight into the behavior of users and processes, it is recommended that the delegation of audit review rights be performed with caution.

Rozdział 17. Storage

17.1. Synopsis

This chapter covers the use of disks in FreeBSD. This includes memory-backed disks, network-attached disks, standard SCSI/IDE storage devices, and devices using the USB interface.

After reading this chapter, you will know:

- The terminology FreeBSD uses to describe the organization of data on a physical disk (partitions and slices).
- How to add additional hard disks to your system.
- How to configure FreeBSD to use USB storage devices.
- How to set up virtual file systems, such as memory disks.
- How to use quotas to limit disk space usage.
- How to encrypt disks to secure them against attackers.
- How to create and burn CDs and DVDs on FreeBSD.
- The various storage media options for backups.
- How to use backup programs available under FreeBSD.
- How to backup to floppy disks.
- What file system snapshots are and how to use them efficiently.

Before reading this chapter, you should:

- Know how to configure and install a new FreeBSD kernel (Rozdział 8).

17.2. Device Names

The following is a list of physical storage devices supported in FreeBSD, and the device names associated with them.

Tabela 17-1. Physical Disk Naming Conventions

Drive type	Drive device name
IDE hard drives	<code>ad</code>
IDE CDROM drives	<code>acd</code>
SCSI hard drives and USB Mass storage devices	<code>da</code>
SCSI CDROM drives	<code>cd</code>
Assorted non-standard CDROM drives	<code>mcd</code> for Mitsumi CD-ROM and <code>scd</code> for Sony CD-ROM devices
Floppy drives	<code>fd</code>
SCSI tape drives	<code>sa</code>
IDE tape drives	<code>ast</code>

Drive type

Flash drives

RAID drives

Drive device name

fla for DiskOnChip® Flash device

aacd for Adaptec AdvancedRAID, mlx0 and mlyd for Mylex, amrd for AMI MegaRAID, idad for Compaq Smart RAID, twed for 3ware® RAID.

17.3. Adding Disks

Originally contributed by David O'Brien.

Lets say we want to add a new SCSI disk to a machine that currently only has a single drive. First turn off the computer and install the drive in the computer following the instructions of the computer, controller, and drive manufacturer. Due to the wide variations of procedures to do this, the details are beyond the scope of this document.

Login as user `root`. After you have installed the drive, inspect `/var/run/dmesg.boot` to ensure the new disk was found. Continuing with our example, the newly added drive will be `da1` and we want to mount it on `/1` (if you are adding an IDE drive, the device name will be `ad1`).

FreeBSD runs on IBM-PC compatible computers, therefore it must take into account the PC BIOS partitions. These are different from the traditional BSD partitions. A PC disk has up to four BIOS partition entries. If the disk is going to be truly dedicated to FreeBSD, you can use the *dedicated* mode. Otherwise, FreeBSD will have to live within one of the PC BIOS partitions. FreeBSD calls the PC BIOS partitions *slices* so as not to confuse them with traditional BSD partitions. You may also use slices on a disk that is dedicated to FreeBSD, but used in a computer that also has another operating system installed. This is a good way to avoid confusing the `fdisk` utility of other, non-FreeBSD operating systems.

In the slice case the drive will be added as `/dev/da1s1e`. This is read as: SCSI disk, unit number 1 (second SCSI disk), slice 1 (PC BIOS partition 1), and `e` BSD partition. In the dedicated case, the drive will be added simply as `/dev/da1e`.

Due to the use of 32-bit integers to store the number of sectors, `bsdlable(8)` is limited to $2^{32}-1$ sectors per disk or 2TB in most cases. The `fdisk(8)` format allows a starting sector of no more than $2^{32}-1$ and a length of no more than $2^{32}-1$, limiting partitions to 2TB and disks to 4TB in most cases. The `sunlabel(8)` format is limited to $2^{32}-1$ sectors per partition and 8 partitions for a total of 16TB. For larger disks, `gpt(8)` partitions may be used.

17.3.1. Using `sysinstall(8)`

1. Navigating `Sysinstall`

You may use `sysinstall` to partition and label a new disk using its easy to use menus. Either login as user `root` or use the `su` command. Run `sysinstall` and enter the Configure menu. Within the FreeBSD Configuration Menu, scroll down and select the `Fdisk` option.

2. `fdisk` Partition Editor

Once inside `fdisk`, typing `A` will use the entire disk for FreeBSD. When asked if you want to „remain cooperative with any future possible operating systems”, answer `YES`. Write the changes to the disk using `w`. Now exit the `FDISK` editor by typing `q`. Next you will be asked about the „Master Boot Record”. Since you are adding a disk to an already running system, choose `None`.

3. Disk Label Editor

Next, you need to exit **sysinstall** and start it again. Follow the directions above, although this time choose the Label option. This will enter the Disk Label Editor. This is where you will create the traditional BSD partitions. A disk can have up to eight partitions, labeled a-h. A few of the partition labels have special uses. The a partition is used for the root partition (/). Thus only your system disk (e.g, the disk you boot from) should have an a partition. The b partition is used for swap partitions, and you may have many disks with swap partitions. The c partition addresses the entire disk in dedicated mode, or the entire FreeBSD slice in slice mode. The other partitions are for general use.

sysinstall's Label editor favors the e partition for non-root, non-swap partitions. Within the Label editor, create a single file system by typing c. When prompted if this will be a FS (file system) or swap, choose FS and type in a mount point (e.g, /mnt). When adding a disk in post-install mode, **sysinstall** will not create entries in /etc/fstab for you, so the mount point you specify is not important.

You are now ready to write the new label to the disk and create a file system on it. Do this by typing w. Ignore any errors from **sysinstall** that it could not mount the new partition. Exit the Label Editor and **sysinstall** completely.

4. Finish

The last step is to edit /etc/fstab to add an entry for your new disk.

17.3.2. Using Command Line Utilities

17.3.2.1. Using Slices

This setup will allow your disk to work correctly with other operating systems that might be installed on your computer and will not confuse other operating systems' fdisk utilities. It is recommended to use this method for new disk installs. Only use dedicated mode if you have a good reason to do so!

```
# dd if=/dev/zero of=/dev/dal bs=1k count=1
# fdisk -BI dal #Initialize your new disk
# bsdlablel -B -w -r dals1 auto #Label it.
# bsdlablel -e dals1 # Edit the bsdlablel just created and add any partitions.
# mkdir -p /1
# newfs /dev/dals1e # Repeat this for every partition you created.
# mount /dev/dals1e /1 # Mount the partition(s)
# vi /etc/fstab # Add the appropriate entry/entries to your /etc/fstab.
```

If you have an IDE disk, substitute ad for da.

17.3.2.2. Dedicated

If you will not be sharing the new drive with another operating system, you may use the dedicated mode. Remember this mode can confuse Microsoft operating systems; however, no damage will be done by them. IBM's OS/2 however, will „appropriate” any partition it finds which it does not understand.

```
# dd if=/dev/zero of=/dev/dal bs=1k count=1
# bsdlablel -Brw dal auto
# bsdlablel -e dal # create the 'e' partition
# newfs -d0 /dev/dale
```

```
# mkdir -p /l
# vi /etc/fstab                # add an entry for /dev/dale
# mount /l
```

An alternate method is:

```
# dd if=/dev/zero of=/dev/dal count=2
# bsdlable /dev/dal | bsdlable -BrR dal /dev/stdin
# newfs /dev/dale
# mkdir -p /l
# vi /etc/fstab                # add an entry for /dev/dale
# mount /l
```

17.4. RAID

17.4.1. Software RAID

17.4.1.1. Concatenated Disk Driver (CCD) Configuration

Original work by Christopher Shumway. Revised by Jim Brown.

When choosing a mass storage solution the most important factors to consider are speed, reliability, and cost. It is rare to have all three in balance; normally a fast, reliable mass storage device is expensive, and to cut back on cost either speed or reliability must be sacrificed.

In designing the system described below, cost was chosen as the most important factor, followed by speed, then reliability. Data transfer speed for this system is ultimately constrained by the network. And while reliability is very important, the CCD drive described below serves online data that is already fully backed up on CD-R's and can easily be replaced.

Defining your own requirements is the first step in choosing a mass storage solution. If your requirements prefer speed or reliability over cost, your solution will differ from the system described in this section.

17.4.1.1.1. Installing the Hardware

In addition to the IDE system disk, three Western Digital 30GB, 5400 RPM IDE disks form the core of the CCD disk described below providing approximately 90GB of online storage. Ideally, each IDE disk would have its own IDE controller and cable, but to minimize cost, additional IDE controllers were not used. Instead the disks were configured with jumpers so that each IDE controller has one master, and one slave.

Upon reboot, the system BIOS was configured to automatically detect the disks attached. More importantly, FreeBSD detected them on reboot:

```
ad0: 19574MB <WDC WD205BA> [39770/16/63] at ata0-master UDMA33
ad1: 29333MB <WDC WD307AA> [59598/16/63] at ata0-slave UDMA33
ad2: 29333MB <WDC WD307AA> [59598/16/63] at ata1-master UDMA33
ad3: 29333MB <WDC WD307AA> [59598/16/63] at ata1-slave UDMA33
```

Notatka: If FreeBSD does not detect all the disks, ensure that you have jumpered them correctly. Most IDE drives also have a „Cable Select” jumper. This is *not* the jumper for the master/slave relationship. Consult the drive documentation for help in identifying the correct jumper.

Next, consider how to attach them as part of the file system. You should research both vinum(8) (Rozdział 19) and ccd(4). In this particular configuration, ccd(4) was chosen.

17.4.1.1.2. Setting Up the CCD

The ccd(4) driver allows you to take several identical disks and concatenate them into one logical file system. In order to use ccd(4), you need a kernel with ccd(4) support built in. Add this line to your kernel configuration file, rebuild, and reinstall the kernel:

```
device    ccd
```

The ccd(4) support can also be loaded as a kernel loadable module.

To set up ccd(4), you must first use bsdlabeled(8) to label the disks:

```
bsdlabeled -r -w ad1c auto
bsdlabeled -r -w ad2c auto
bsdlabeled -r -w ad3c auto
```

This creates a bsdlabeled for ad1c, ad2c and ad3c that spans the entire disk.

The next step is to change the disk label type. You can use bsdlabeled(8) to edit the disks:

```
bsdlabeled -e ad1
bsdlabeled -e ad2
bsdlabeled -e ad3
```

This opens up the current disk label on each disk with the editor specified by the EDITOR environment variable, typically vi(1).

An unmodified disk label will look something like this:

```
8 partitions:
#          size      offset      fstype    [fsize bsize bps/cpg]
  c: 60074784         0      unused          0      0      0   # (Cyl.    0 - 59597)
```

Add a new e partition for ccd(4) to use. This can usually be copied from the c partition, but the fstype *must* be **4.2BSD**. The disk label should now look something like this:

```
8 partitions:
#          size      offset      fstype    [fsize bsize bps/cpg]
  c: 60074784         0      unused          0      0      0   # (Cyl.    0 - 59597)
  e: 60074784         0      4.2BSD          0      0      0   # (Cyl.    0 - 59597)
```

17.4.1.1.3. Building the File System

Now that you have all the disks labeled, you must build the ccd(4). To do that, use ccdconfig(8), with options similar to the following:

```
ccdconfig ccd0❶ 32❷ 0❸ /dev/ad1e❹ /dev/ad2e /dev/ad3e
```

The use and meaning of each option is shown below:

- ❶ The first argument is the device to configure, in this case, /dev/ccd0c. The /dev/ portion is optional.
- ❷ The interleave for the file system. The interleave defines the size of a stripe in disk blocks, each normally 512 bytes. So, an interleave of 32 would be 16,384 bytes.
- ❸ Flags for ccdconfig(8). If you want to enable drive mirroring, you can specify a flag here. This configuration does not provide mirroring for ccd(4), so it is set at 0 (zero).
- ❹ The final arguments to ccdconfig(8) are the devices to place into the array. Use the complete pathname for each device.

After running ccdconfig(8) the ccd(4) is configured. A file system can be installed. Refer to newfs(8) for options, or simply run:

```
newfs /dev/ccd0c
```

17.4.1.1.4. Making it All Automatic

Generally, you will want to mount the ccd(4) upon each reboot. To do this, you must configure it first. Write out your current configuration to /etc/ccd.conf using the following command:

```
ccdconfig -g > /etc/ccd.conf
```

During reboot, the script /etc/rc runs ccdconfig -C if /etc/ccd.conf exists. This automatically configures the ccd(4) so it can be mounted.

Notatka: If you are booting into single user mode, before you can mount(8) the ccd(4), you need to issue the following command to configure the array:

```
ccdconfig -C
```

To automatically mount the ccd(4), place an entry for the ccd(4) in /etc/fstab so it will be mounted at boot time:

```
/dev/ccd0c          /media             ufs                rw                2                2
```

17.4.1.2. The Vinum Volume Manager

The Vinum Volume Manager is a block device driver which implements virtual disk drives. It isolates disk hardware from the block device interface and maps data in ways which result in an increase in flexibility, performance and reliability compared to the traditional slice view of disk storage. vinum(8) implements the RAID-0, RAID-1 and RAID-5 models, both individually and in combination.

See Rozdział 19 for more information about vinum(8).

17.4.2. Hardware RAID

FreeBSD also supports a variety of hardware RAID controllers. These devices control a RAID subsystem without the need for FreeBSD specific software to manage the array.

Using an on-card BIOS, the card controls most of the disk operations itself. The following is a brief setup description using a Promise IDE RAID controller. When this card is installed and the system is started up, it displays a prompt requesting information. Follow the instructions to enter the card's setup screen. From here, you have the ability to combine all the attached drives. After doing so, the disk(s) will look like a single drive to FreeBSD. Other RAID levels can be set up accordingly.

17.4.3. Rebuilding ATA RAID1 Arrays

FreeBSD allows you to hot-replace a failed disk in an array. This requires that you catch it before you reboot.

You will probably see something like the following in `/var/log/messages` or in the `dmesg(8)` output:

```
ad6 on monster1 suffered a hard error.
ad6: READ command timeout tag=0 serv=0 - resetting
ad6: trying fallback to PIO mode
ata3: resetting devices .. done
ad6: hard error reading fsbn 1116119 of 0-7 (ad6 bn 1116119; cn 1107 tn 4 sn 11)\\
status=59 error=40
ar0: WARNING - mirror lost
```

Using `atacontrol(8)`, check for further information:

```
# atactrol list
ATA channel 0:
    Master:      no device present
    Slave:      acd0 <HL-DT-ST CD-ROM GCR-8520B/1.00> ATA/ATAPI rev 0

ATA channel 1:
    Master:      no device present
    Slave:      no device present

ATA channel 2:
    Master:      ad4 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:      no device present

ATA channel 3:
    Master:      ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
    Slave:      no device present

# atactrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: DEGRADED
```

1. You will first need to detach the ata channel with the failed disk so you can safely remove it:

```
# atacontrol detach ata3
```

2. Replace the disk.

3. Reattach the ata channel:

```
# atacontrol attach ata3
```

```
Master:  ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
```

```
Slave:   no device present
```

4. Add the new disk to the array as a spare:

```
# atacontrol addspare ar0 ad6
```

5. Rebuild the array:

```
# atacontrol rebuild ar0
```

6. It is possible to check on the progress by issuing the following command:

```
# dmesg | tail -10
```

```
[output removed]
```

```
ad6: removed from configuration
```

```
ad6: deleted from ar0 disk1
```

```
ad6: inserted into ar0 disk1 as spare
```

```
# atacontrol status ar0
```

```
ar0: ATA RAID1 subdisks: ad4 ad6 status: REBUILDING 0% completed
```

7. Wait until this operation completes.

17.5. USB Storage Devices

Contributed by Marc Fonvieille.

A lot of external storage solutions, nowadays, use the Universal Serial Bus (USB): hard drives, USB thumbdrives, CD-R burners, etc. FreeBSD provides support for these devices.

17.5.1. Configuration

The USB mass storage devices driver, `umass(4)`, provides the support for USB storage devices. If you use the `GENERIC` kernel, you do not have to change anything in your configuration. If you use a custom kernel, be sure that the following lines are present in your kernel configuration file:

```
device scbus
```

```
device da
```

```
device pass
```

```
device uhci
```

```
device ohci
```

```
device usb
```

```
device umass
```

The `umass(4)` driver uses the SCSI subsystem to access to the USB storage devices, your USB device will be seen as a SCSI device by the system. Depending on the USB chipset on your motherboard, you only need either `device`

uhci or device ohci, however having both in the kernel configuration file is harmless. Do not forget to compile and install the new kernel if you added any lines.

Notatka: If your USB device is a CD-R or DVD burner, the SCSI CD-ROM driver, cd(4), must be added to the kernel via the line:

```
device cd
```

Since the burner is seen as a SCSI drive, the driver atapicam(4) should not be used in the kernel configuration.

Support for USB 2.0 controllers is provided on FreeBSD; however, you must add:

```
device ehci
```

to your configuration file for USB 2.0 support. Note uhci(4) and ohci(4) drivers are still needed if you want USB 1.X support.

17.5.2. Testing the Configuration

The configuration is ready to be tested: plug in your USB device, and in the system message buffer (dmesg(8)), the drive should appear as something like:

```
umass0: USB Solid state disk, rev 1.10/1.00, addr 2
GEOM: create disk da0 dp=0xc2d74850
da0 at umass-sim0 bus 0 target 0 lun 0
da0: <Generic Traveling Disk 1.11> Removable Direct Access SCSI-2 device
da0: 1.000MB/s transfers
da0: 126MB (258048 512 byte sectors: 64H 32S/T 126C)
```

Of course, the brand, the device node (da0) and other details can differ according to your configuration.

Since the USB device is seen as a SCSI one, the camcontrol command can be used to list the USB storage devices attached to the system:

```
# camcontrol devlist
<Generic Traveling Disk 1.11>          at scbus0 target 0 lun 0 (da0,pass0)
```

If the drive comes with a file system, you should be able to mount it. The Sekcja 17.3 will help you to format and create partitions on the USB drive if needed.

If you unplug the device (the disk must be unmounted before), you should see, in the system message buffer, something like the following:

```
umass0: at uhub0 port 1 (addr 2) disconnected
(da0:umass-sim0:0:0:0): lost device
(da0:umass-sim0:0:0:0): removing device entry
GEOM: destroy disk da0 dp=0xc2d74850
umass0: detached
```

17.5.3. Further Reading

Beside the Adding Disks and Mounting and Unmounting File Systems sections, reading various manual pages may be also useful: `umass(4)`, `camcontrol(8)`, and `usbdevs(8)`.

17.6. Creating and Using Optical Media (CDs)

Contributed by Mike Meyer.

17.6.1. Introduction

CDs have a number of features that differentiate them from conventional disks. Initially, they were not writable by the user. They are designed so that they can be read continuously without delays to move the head between tracks. They are also much easier to transport between systems than similarly sized media were at the time.

CDs do have tracks, but this refers to a section of data to be read continuously and not a physical property of the disk. To produce a CD on FreeBSD, you prepare the data files that are going to make up the tracks on the CD, then write the tracks to the CD.

The ISO 9660 file system was designed to deal with these differences. It unfortunately codifies file system limits that were common then. Fortunately, it provides an extension mechanism that allows properly written CDs to exceed those limits while still working with systems that do not support those extensions.

The `sysutils/cdrtools` port includes `mkisofs(8)`, a program that you can use to produce a data file containing an ISO 9660 file system. It has options that support various extensions, and is described below.

Which tool to use to burn the CD depends on whether your CD burner is ATAPI or something else. ATAPI CD burners use the `burncd` program that is part of the base system. SCSI and USB CD burners should use `cdrecord` from the `sysutils/cdrtools` port. It is also possible to use `cdrecord` and other tools for SCSI drives on ATAPI hardware with the ATAPI/CAM module.

If you want CD burning software with a graphical user interface, you may wish to take a look at either **X-CD-Roast** or **K3b**. These tools are available as packages or from the `sysutils/xcdroast` and `sysutils/k3b` ports. **X-CD-Roast** and **K3b** require the ATAPI/CAM module with ATAPI hardware.

17.6.2. mkisofs

The `mkisofs(8)` program, which is part of the `sysutils/cdrtools` port, produces an ISO 9660 file system that is an image of a directory tree in the UNIX file system name space. The simplest usage is:

```
# mkisofs -o imagefile.iso /path/to/tree
```

This command will create an `imagefile.iso` containing an ISO 9660 file system that is a copy of the tree at `/path/to/tree`. In the process, it will map the file names to names that fit the limitations of the standard ISO 9660 file system, and will exclude files that have names uncharacteristic of ISO file systems.

A number of options are available to overcome those restrictions. In particular, `-R` enables the Rock Ridge extensions common to UNIX systems, `-J` enables Joliet extensions used by Microsoft systems, and `-hfs` can be used to create HFS file systems used by Mac OS.

For CDs that are going to be used only on FreeBSD systems, `-U` can be used to disable all filename restrictions. When used with `-R`, it produces a file system image that is identical to the FreeBSD tree you started from, though it may violate the ISO 9660 standard in a number of ways.

The last option of general use is `-b`. This is used to specify the location of the boot image for use in producing an „El Torito” bootable CD. This option takes an argument which is the path to a boot image from the top of the tree being written to the CD. By default, `mkisofs(8)` creates an ISO image in the so-called „floppy disk emulation” mode, and thus expects the boot image to be exactly 1200, 1440 or 2880 KB in size. Some boot loaders, like the one used by the FreeBSD distribution disks, do not use emulation mode; in this case, the `-no-emul-boot` option should be used. So, if `/tmp/myboot` holds a bootable FreeBSD system with the boot image in `/tmp/myboot/boot/cdboot`, you could produce the image of an ISO 9660 file system in `/tmp/bootable.iso` like so:

```
# mkisofs -R -no-emul-boot -b boot/cdboot -o /tmp/bootable.iso /tmp/myboot
```

Having done that, if you have `md` configured in your kernel, you can mount the file system with:

```
# mdconfig -a -t vnode -f /tmp/bootable.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

At which point you can verify that `/mnt` and `/tmp/myboot` are identical.

There are many other options you can use with `mkisofs(8)` to fine-tune its behavior. In particular: modifications to an ISO 9660 layout and the creation of Joliet and HFS discs. See the `mkisofs(8)` manual page for details.

17.6.3. burncd

If you have an ATAPI CD burner, you can use the `burncd` command to burn an ISO image onto a CD. `burncd` is part of the base system, installed as `/usr/sbin/burncd`. Usage is very simple, as it has few options:

```
# burncd -f cddevice data imagefile.iso fixate
```

Will burn a copy of `imagefile.iso` on `cddevice`. The default device is `/dev/acd0`. See `burncd(8)` for options to set the write speed, eject the CD after burning, and write audio data.

17.6.4. cdrecord

If you do not have an ATAPI CD burner, you will have to use `cdrecord` to burn your CDs. `cdrecord` is not part of the base system; you must install it from either the port at `sysutils/cdrtools` or the appropriate package. Changes to the base system can cause binary versions of this program to fail, possibly resulting in a „coaster”. You should therefore either upgrade the port when you upgrade your system, or if you are tracking `-STABLE`, upgrade the port when a new version becomes available.

While `cdrecord` has many options, basic usage is even simpler than `burncd`. Burning an ISO 9660 image is done with:

```
# cdrecord dev=device imagefile.iso
```

The tricky part of using `cdrecord` is finding the `dev` to use. To find the proper setting, use the `-scanbus` flag of `cdrecord`, which might produce results like this:

```
# cdrecord -scanbus
```

```
Cdrecord-Clone 2.01 (i386-unknown-freebsd7.0) Copyright (C) 1995-2004 Jörg Schilling
Using libscg version 'schily-0.1'
```

```
scsibus0:
  0,0,0      0) 'SEAGATE ' 'ST39236LW      ' '0004' Disk
  0,1,0      1) 'SEAGATE ' 'ST39173W      ' '5958' Disk
  0,2,0      2) *
  0,3,0      3) 'iomega ' 'jaz 1GB        ' 'J.86' Removable Disk
  0,4,0      4) 'NEC      ' 'CD-ROM DRIVE:466' '1.26' Removable CD-ROM
  0,5,0      5) *
  0,6,0      6) *
  0,7,0      7) *

scsibus1:
  1,0,0     100) *
  1,1,0     101) *
  1,2,0     102) *
  1,3,0     103) *
  1,4,0     104) *
  1,5,0     105) 'YAMAHA ' 'CRW4260        ' '1.0q' Removable CD-ROM
  1,6,0     106) 'ARTEC  ' 'AM12S          ' '1.06' Scanner
  1,7,0     107) *
```

This lists the appropriate dev value for the devices on the list. Locate your CD burner, and use the three numbers separated by commas as the value for `dev`. In this case, the CRW device is 1,5,0, so the appropriate input would be `dev=1,5,0`. There are easier ways to specify this value; see `cdrecord(1)` for details. That is also the place to look for information on writing audio tracks, controlling the speed, and other things.

17.6.5. Duplicating Audio CDs

You can duplicate an audio CD by extracting the audio data from the CD to a series of files, and then writing these files to a blank CD. The process is slightly different for ATAPI and SCSI drives.

SCSI Drives

1. Use `cdda2wav` to extract the audio.

```
% cdda2wav -v255 -D2,0 -B -Owav
```

2. Use `cdrecord` to write the `.wav` files.

```
% cdrecord -v dev=2,0 -dao -useinfo *.wav
```

Make sure that `2,0` is set appropriately, as described in Sekcja 17.6.4.

ATAPI Drives

1. The ATAPI CD driver makes each track available as `/dev/acd0t nn` , where d is the drive number, and nn is the track number written with two decimal digits, prefixed with zero as needed. So the first track on the first disk is `/dev/acd0t01`, the second is `/dev/acd0t02`, the third is `/dev/acd0t03`, and so on.

Make sure the appropriate files exist in `/dev`. If the entries are missing, force the system to retaste the media:

```
# dd if=/dev/acd0 of=/dev/null count=1
```

2. Extract each track using `dd(1)`. You must also use a specific block size when extracting the files.

```
# dd if=/dev/acd0t01 of=track1.cdr bs=2352
# dd if=/dev/acd0t02 of=track2.cdr bs=2352
...
```

3. Burn the extracted files to disk using `burncd`. You must specify that these are audio files, and that `burncd` should fixate the disk when finished.

```
# burncd -f /dev/acd0 audio track1.cdr track2.cdr ... fixate
```

17.6.6. Duplicating Data CDs

You can copy a data CD to a image file that is functionally equivalent to the image file created with `mkisofs(8)`, and you can use it to duplicate any data CD. The example given here assumes that your CDROM device is `acd0`. Substitute your correct CDROM device.

```
# dd if=/dev/acd0 of=file.iso bs=2048
```

Now that you have an image, you can burn it to CD as described above.

17.6.7. Using Data CDs

Now that you have created a standard data CDROM, you probably want to mount it and read the data on it. By default, `mount(8)` assumes that a file system is of type `ufs`. If you try something like:

```
# mount /dev/cd0 /mnt
```

you will get a complaint about `Incorrect super block`, and no mount. The CDROM is not a `ufs` file system, so attempts to mount it as such will fail. You just need to tell `mount(8)` that the file system is of type `ISO9660`, and everything will work. You do this by specifying the `-t cd9660` option `mount(8)`. For example, if you want to mount the CDROM device, `/dev/cd0`, under `/mnt`, you would execute:

```
# mount -t cd9660 /dev/cd0 /mnt
```

Note that your device name (`/dev/cd0` in this example) could be different, depending on the interface your CDROM uses. Also, the `-t cd9660` option just executes `mount_cd9660(8)`. The above example could be shortened to:

```
# mount_cd9660 /dev/cd0 /mnt
```

You can generally use data CDROMs from any vendor in this way. Disks with certain ISO 9660 extensions might behave oddly, however. For example, Joliet disks store all filenames in two-byte Unicode characters. The FreeBSD kernel does not speak Unicode (yet!), so non-English characters show up as question marks. (The FreeBSD CD9660 driver includes hooks to load an appropriate Unicode conversion table on the fly. Modules for some of the common encodings are available via the `sysutils/cd9660_unicode` port.)

Occasionally, you might get `Device not configured` when trying to mount a CDROM. This usually means that the CDROM drive thinks that there is no disk in the tray, or that the drive is not visible on the bus. It can take a couple of seconds for a CDROM drive to realize that it has been fed, so be patient.

Sometimes, a SCSI CDROM may be missed because it did not have enough time to answer the bus reset. If you have a SCSI CDROM please add the following option to your kernel configuration and rebuild your kernel.

```
options SCSI_DELAY=15000
```

This tells your SCSI bus to pause 15 seconds during boot, to give your CDROM drive every possible chance to answer the bus reset.

17.6.8. Burning Raw Data CDs

You can choose to burn a file directly to CD, without creating an ISO 9660 file system. Some people do this for backup purposes. This runs more quickly than burning a standard CD:

```
# burncd -f /dev/acd1 -s 12 data archive.tar.gz fixate
```

In order to retrieve the data burned to such a CD, you must read data from the raw device node:

```
# tar xzvf /dev/acd1
```

You cannot mount this disk as you would a normal CDROM. Such a CDROM cannot be read under any operating system except FreeBSD. If you want to be able to mount the CD, or share data with another operating system, you must use mkisofs(8) as described above.

17.6.9. Using the ATAPI/CAM Driver

Contributed by Marc Fonvieille.

This driver allows ATAPI devices (CD-ROM, CD-RW, DVD drives etc...) to be accessed through the SCSI subsystem, and so allows the use of applications like sysutils/cdrdao or cdrecord(1).

To use this driver, you will need to add the following line to the `/boot/loader.conf` file:

```
atapicam_load="YES"
```

then, reboot your machine.

Notatka: If you prefer to statically compile the atapicam(4) support in your kernel, you will have to add this line to your kernel configuration file:

```
device atapicam
```

You also need the following lines in your kernel configuration file:

```
device ata
device scbus
device cd
device pass
```

which should already be present. Then rebuild, install your new kernel, and reboot your machine.

During the boot process, your burner should show up, like so:

```
acd0: CD-RW <MATSHITA CD-RW/DVD-ROM UJDA740> at atal-master PIO4
cd0 at atal bus 0 target 0 lun 0
cd0: <MATSHITA CDRW/DVD UJDA740 1.00> Removable CD-ROM SCSI-0 device
cd0: 16.000MB/s transfers
```

```
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

The drive could now be accessed via the `/dev/cd0` device name, for example to mount a CD-ROM on `/mnt`, just type the following:

```
# mount -t cd9660 /dev/cd0 /mnt
```

As `root`, you can run the following command to get the SCSI address of the burner:

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00> at scbus1 target 0 lun 0 (pass0,cd0)
```

So `1,0,0` will be the SCSI address to use with `cdrecord(1)` and other SCSI application.

For more information about ATAPI/CAM and SCSI system, refer to the `atapicam(4)` and `cam(4)` manual pages.

17.7. Creating and Using Optical Media (DVDs)

Contributed by Marc Fonvieille. With inputs from Andy Polyakov.

17.7.1. Introduction

Compared to the CD, the DVD is the next generation of optical media storage technology. The DVD can hold more data than any CD and is nowadays the standard for video publishing.

Five physical recordable formats can be defined for what we will call a recordable DVD:

- **DVD-R:** This was the first DVD recordable format available. The DVD-R standard is defined by the DVD Forum (<http://www.dvdforum.com/forum.shtml>). This format is write once.
- **DVD-RW:** This is the rewritable version of the DVD-R standard. A DVD-RW can be rewritten about 1000 times.
- **DVD-RAM:** This is also a rewritable format supported by the DVD Forum. A DVD-RAM can be seen as a removable hard drive. However, this media is not compatible with most DVD-ROM drives and DVD-Video players; only a few DVD writers support the DVD-RAM format. Read the *Sekcja 17.7.9* for more information on DVD-RAM use.
- **DVD+RW:** This is a rewritable format defined by the DVD+RW Alliance (<http://www.dvdrw.com/>). A DVD+RW can be rewritten about 1000 times.
- **DVD+R:** This format is the write once variation of the DVD+RW format.

A single layer recordable DVD can hold up to 4,700,000,000 bytes which is actually 4.38 GB or 4485 MB (1 kilobyte is 1024 bytes).

Notatka: A distinction must be made between the physical media and the application. For example, a DVD-Video is a specific file layout that can be written on any recordable DVD physical media: DVD-R, DVD+R, DVD-RW etc. Before choosing the type of media, you must be sure that both the burner and the DVD-Video player (a standalone player or a DVD-ROM drive on a computer) are compatible with the media under consideration.

17.7.2. Configuration

The program `growisofs(1)` will be used to perform DVD recording. This command is part of the **dvd+rw-tools** utilities (`sysutils/dvd+rw-tools`). The **dvd+rw-tools** support all DVD media types.

These tools use the SCSI subsystem to access to the devices, therefore the ATAPI/CAM support must be added to your kernel. If your burner uses the USB interface this addition is useless, and you should read the Sekcja 17.5 for more details on USB devices configuration.

You also have to enable DMA access for ATAPI devices, this can be done in adding the following line to the `/boot/loader.conf` file:

```
hw.ata.atapi_dma="1"
```

Before attempting to use the **dvd+rw-tools** you should consult the `dvd+rw-tools`' hardware compatibility notes (<http://fy.chalmers.se/~appro/linux/DVD+RW/hcn.html>) for any information related to your DVD burner.

Notatka: If you want a graphical user interface, you should have a look to **K3b** (`sysutils/k3b`) which provides a user friendly interface to `growisofs(1)` and many others burning tools.

17.7.3. Burning Data DVDs

The `growisofs(1)` command is a frontend to `mkisofs(8)`, it will invoke `mkisofs(8)` to create the file system layout and will perform the write on the DVD. This means you do not need to create an image of the data before the burning process.

To burn onto a DVD+R or a DVD-R the data from the `/path/to/data` directory, use the following command:

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /path/to/data
```

The options `-J -R` are passed to `mkisofs(8)` for the file system creation (in this case: an ISO 9660 file system with Joliet and Rock Ridge extensions), consult the `mkisofs(8)` manual page for more details.

The option `-Z` is used for the initial session recording in any case: multiple sessions or not. The DVD device, `/dev/cd0`, must be changed according to your configuration. The `-dvd-compat` parameter will close the disk, the recording will be unappendable. In return this should provide better media compatibility with DVD-ROM drives.

It is also possible to burn a pre-mastered image, for example to burn the image `imagefile.iso`, we will run:

```
# growisofs -dvd-compat -Z /dev/cd0=imagefile.iso
```

The write speed should be detected and automatically set according to the media and the drive being used. If you want to force the write speed, use the `-speed=` parameter. For more information, read the `growisofs(1)` manual page.

17.7.4. Burning a DVD-Video

A DVD-Video is a specific file layout based on ISO 9660 and the micro-UDF (M-UDF) specifications. The DVD-Video also presents a specific data structure hierarchy, it is the reason why you need a particular program such as `multimedia/dvdauthor` to author the DVD.

If you already have an image of the DVD-Video file system, just burn it in the same way as for any image, see the previous section for an example. If you have made the DVD authoring and the result is in, for example, the directory `/path/to/video`, the following command should be used to burn the DVD-Video:

```
# growisofs -Z /dev/cd0 -dvd-video /path/to/video
```

The `-dvd-video` option will be passed down to `mkisofs(8)` and will instruct it to create a DVD-Video file system layout. Beside this, the `-dvd-video` option implies `-dvd-compatible growisofs(1)` option.

17.7.5. Using a DVD+RW

Unlike CD-RW, a virgin DVD+RW needs to be formatted before first use. The `growisofs(1)` program will take care of it automatically whenever appropriate, which is the *recommended* way. However you can use the `dvd+rw-format` command to format the DVD+RW:

```
# dvd+rw-format /dev/cd0
```

You need to perform this operation just once, keep in mind that only virgin DVD+RW medias need to be formatted. Then you can burn the DVD+RW in the way seen in previous sections.

If you want to burn new data (burn a totally new file system not append some data) onto a DVD+RW, you do not need to blank it, you just have to write over the previous recording (in performing a new initial session), like this:

```
# growisofs -Z /dev/cd0 -J -R /path/to/newdata
```

DVD+RW format offers the possibility to easily append data to a previous recording. The operation consists in merging a new session to the existing one, it is not multisession writing, `growisofs(1)` will *grow* the ISO 9660 file system present on the media.

For example, if we want to append data to our previous DVD+RW, we have to use the following:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

The same `mkisofs(8)` options we used to burn the initial session should be used during next writes.

Notatka: You may want to use the `-dvd-compatible` option if you want better media compatibility with DVD-ROM drives. In the DVD+RW case, this will not prevent you from adding data.

If for any reason you really want to blank the media, do the following:

```
# growisofs -Z /dev/cd0=/dev/zero
```

17.7.6. Using a DVD-RW

A DVD-RW accepts two disc formats: the incremental sequential one and the restricted overwrite. By default DVD-RW discs are in sequential format.

A virgin DVD-RW can be directly written without the need of a formatting operation, however a non-virgin DVD-RW in sequential format needs to be blanked before to be able to write a new initial session.

To blank a DVD-RW in sequential mode, run:

```
# dvd+rw-format -blank=full /dev/cd0
```

Notatka: A full blanking (`-blank=full`) will take about one hour on a 1x media. A fast blanking can be performed using the `-blank` option if the DVD-RW will be recorded in Disk-At-Once (DAO) mode. To burn the DVD-RW in DAO mode, use the command:

```
# growisofs -use-the-force-luke=dao -Z /dev/cd0=imagefile.iso
```

The `-use-the-force-luke=dao` option should not be required since `growisofs(1)` attempts to detect minimally (fast blanked) media and engage DAO write.

In fact one should use restricted overwrite mode with any DVD-RW, this format is more flexible than the default incremental sequential one.

To write data on a sequential DVD-RW, use the same instructions as for the other DVD formats:

```
# growisofs -Z /dev/cd0 -J -R /path/to/data
```

If you want to append some data to your previous recording, you will have to use the `growisofs(1)` `-M` option. However, if you perform data addition on a DVD-RW in incremental sequential mode, a new session will be created on the disc and the result will be a multi-session disc.

A DVD-RW in restricted overwrite format does not need to be blanked before a new initial session, you just have to overwrite the disc with the `-Z` option, this is similar to the DVD+RW case. It is also possible to grow an existing ISO 9660 file system written on the disc in a same way as for a DVD+RW with the `-M` option. The result will be a one-session DVD.

To put a DVD-RW in the restricted overwrite format, the following command must be used:

```
# dvd+rw-format /dev/cd0
```

To change back to the sequential format use:

```
# dvd+rw-format -blank=full /dev/cd0
```

17.7.7. Multisession

Very few DVD-ROM drives support multisession DVDs, they will most of time, hopefully, only read the first session. DVD+R, DVD-R and DVD-RW in sequential format can accept multiple sessions, the notion of multiple sessions does not exist for the DVD+RW and the DVD-RW restricted overwrite formats.

Using the following command after an initial (non-closed) session on a DVD+R, DVD-R, or DVD-RW in sequential format, will add a new session to the disc:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

Using this command line with a DVD+RW or a DVD-RW in restricted overwrite mode, will append data in merging the new session to the existing one. The result will be a single-session disc. This is the way used to add data after an initial write on these medias.

Notatka: Some space on the media is used between each session for end and start of sessions. Therefore, one should add sessions with large amount of data to optimize media space. The number of sessions is limited to 154 for a DVD+R, about 2000 for a DVD-R, and 127 for a DVD+R Double Layer.

17.7.8. For More Information

To obtain more information about a DVD, the `dvd+rw-mediainfo /dev/cd0` command can be ran with the disc in the drive.

More information about the **dvd+rw-tools** can be found in the `growisofs(1)` manual page, on the `dvd+rw-tools` web site (<http://fy.chalmers.se/~appro/linux/DVD+RW/>) and in the `cdwrite` mailing list (<http://lists.debian.org/cdwrite/>) archives.

Notatka: The `dvd+rw-mediainfo` output of the resulting recording or the media with issues is mandatory for any problem report. Without this output, it will be quite impossible to help you.

17.7.9. Using a DVD-RAM

17.7.9.1. Configuration

DVD-RAM writers come with either SCSI or ATAPI interface. DMA access for ATAPI devices has to be enabled, this can be done by adding the following line to the `/boot/loader.conf` file:

```
hw.ata.atapi_dma="1"
```

17.7.9.2. Preparing the Medium

As previously mentioned in the chapter introduction, a DVD-RAM can be seen as a removable hard drive. As any other hard drive the DVD-RAM must be „prepared” before the first use. In the example, the whole disk space will be used with a standard UFS2 file system:

```
# dd if=/dev/zero of=/dev/acd0 count=2
# bsdlabel -Bw acd0
# newfs /dev/acd0
```

The DVD device, `acd0`, must be changed according to the configuration.

17.7.9.3. Using the Medium

Once the previous operations have been performed on the DVD-RAM, it can be mounted as a normal hard drive:

```
# mount /dev/acd0 /mnt
```

After this the DVD-RAM will be both readable and writeable.

17.8. Creating and Using Floppy Disks

Original work by Julio Merino. Rewritten by Martin Karlsson.

Storing data on floppy disks is sometimes useful, for example when one does not have any other removable storage media or when one needs to transfer small amounts of data to another computer.

This section will explain how to use floppy disks in FreeBSD. It will primarily cover formatting and usage of 3.5inch DOS floppies, but the concepts are similar for other floppy disk formats.

17.8.1. Formatting Floppies

17.8.1.1. The Device

Floppy disks are accessed through entries in `/dev`, just like other devices. To access the raw floppy disk, simply use `/dev/fdN`.

17.8.1.2. Formatting

A floppy disk needs to be low-level formatted before it can be used. This is usually done by the vendor, but formatting is a good way to check media integrity. Although it is possible to force larger (or smaller) disk sizes, 1440kB is what most floppy disks are designed for.

To low-level format the floppy disk you need to use `fdformat(1)`. This utility expects the device name as an argument. Make note of any error messages, as these can help determine if the disk is good or bad.

17.8.1.2.1. Formatting Floppy Disks

Use the `/dev/fdN` devices to format the floppy. Insert a new 3.5inch floppy disk in your drive and issue:

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

17.8.2. The Disk Label

After low-level formatting the disk, you will need to place a disk label on it. This disk label will be destroyed later, but it is needed by the system to determine the size of the disk and its geometry later.

The new disk label will take over the whole disk, and will contain all the proper information about the geometry of the floppy. The geometry values for the disk label are listed in `/etc/disktab`.

You can run now `bsdlabel(8)` like so:

```
# /sbin/bsdlabel -B -r -w /dev/fd0 fd1440
```

17.8.3. The File System

Now the floppy is ready to be high-level formatted. This will place a new file system on it, which will let FreeBSD read and write to the disk. After creating the new file system, the disk label is destroyed, so if you want to reformat the disk, you will have to recreate the disk label.

The floppy's file system can be either UFS or FAT. FAT is generally a better choice for floppies.

To put a new file system on the floppy, issue:

```
# /sbin/newfs_msdos /dev/fd0
```

The disk is now ready for use.

17.8.4. Using the Floppy

To use the floppy, mount it with `mount_msdosfs(8)`. One can also use `emulators/mttools` from the ports collection.

17.9. Creating and Using Data Tapes

The major tape media are the 4mm, 8mm, QIC, mini-cartridge and DLT.

17.9.1. 4mm (DDS: Digital Data Storage)

4mm tapes are replacing QIC as the workstation backup media of choice. This trend accelerated greatly when Conner purchased Archive, a leading manufacturer of QIC drives, and then stopped production of QIC drives. 4mm drives are small and quiet but do not have the reputation for reliability that is enjoyed by 8mm drives. The cartridges are less expensive and smaller (3 x 2 x 0.5 inches, 76 x 51 x 12 mm) than 8mm cartridges. 4mm, like 8mm, has comparatively short head life for the same reason, both use helical scan.

Data throughput on these drives starts ~150 kB/s, peaking at ~500 kB/s. Data capacity starts at 1.3 GB and ends at 2.0 GB. Hardware compression, available with most of these drives, approximately doubles the capacity. Multi-drive tape library units can have 6 drives in a single cabinet with automatic tape changing. Library capacities reach 240 GB.

The DDS-3 standard now supports tape capacities up to 12 GB (or 24 GB compressed).

4mm drives, like 8mm drives, use helical-scan. All the benefits and drawbacks of helical-scan apply to both 4mm and 8mm drives.

Tapes should be retired from use after 2,000 passes or 100 full backups.

17.9.2. 8mm (Exabyte)

8mm tapes are the most common SCSI tape drives; they are the best choice of exchanging tapes. Nearly every site has an Exabyte 2 GB 8mm tape drive. 8mm drives are reliable, convenient and quiet. Cartridges are inexpensive and small (4.8 x 3.3 x 0.6 inches; 122 x 84 x 15 mm). One downside of 8mm tape is relatively short head and tape life due to the high rate of relative motion of the tape across the heads.

Data throughput ranges from ~250 kB/s to ~500 kB/s. Data sizes start at 300 MB and go up to 7 GB. Hardware compression, available with most of these drives, approximately doubles the capacity. These drives are available as single units or multi-drive tape libraries with 6 drives and 120 tapes in a single cabinet. Tapes are changed automatically by the unit. Library capacities reach 840+ GB.

The Exabyte „Mammoth” model supports 12 GB on one tape (24 GB with compression) and costs approximately twice as much as conventional tape drives.

Data is recorded onto the tape using helical-scan, the heads are positioned at an angle to the media (approximately 6 degrees). The tape wraps around 270 degrees of the spool that holds the heads. The spool spins while the tape slides over the spool. The result is a high density of data and closely packed tracks that angle across the tape from one edge to the other.

17.9.3. QIC

QIC-150 tapes and drives are, perhaps, the most common tape drive and media around. QIC tape drives are the least expensive „serious” backup drives. The downside is the cost of media. QIC tapes are expensive compared to 8mm or 4mm tapes, up to 5 times the price per GB data storage. But, if your needs can be satisfied with a half-dozen tapes, QIC may be the correct choice. QIC is the *most* common tape drive. Every site has a QIC drive of some density or another. Therein lies the rub, QIC has a large number of densities on physically similar (sometimes identical) tapes. QIC drives are not quiet. These drives audibly seek before they begin to record data and are clearly audible whenever reading, writing or seeking. QIC tapes measure (6 x 4 x 0.7 inches; 152 x 102 x 17 mm).

Data throughput ranges from ~150 kB/s to ~500 kB/s. Data capacity ranges from 40 MB to 15 GB. Hardware compression is available on many of the newer QIC drives. QIC drives are less frequently installed; they are being supplanted by DAT drives.

Data is recorded onto the tape in tracks. The tracks run along the long axis of the tape media from one end to the other. The number of tracks, and therefore the width of a track, varies with the tape’s capacity. Most if not all newer drives provide backward-compatibility at least for reading (but often also for writing). QIC has a good reputation regarding the safety of the data (the mechanics are simpler and more robust than for helical scan drives).

Tapes should be retired from use after 5,000 backups.

17.9.4. DLT

DLT has the fastest data transfer rate of all the drive types listed here. The 1/2" (12.5mm) tape is contained in a single spool cartridge (4 x 4 x 1 inches; 100 x 100 x 25 mm). The cartridge has a swinging gate along one entire side of the cartridge. The drive mechanism opens this gate to extract the tape leader. The tape leader has an oval hole in it which the drive uses to „hook” the tape. The take-up spool is located inside the tape drive. All the other tape cartridges listed here (9 track tapes are the only exception) have both the supply and take-up spools located inside the tape cartridge itself.

Data throughput is approximately 1.5 MB/s, three times the throughput of 4mm, 8mm, or QIC tape drives. Data capacities range from 10 GB to 20 GB for a single drive. Drives are available in both multi-tape changers and multi-tape, multi-drive tape libraries containing from 5 to 900 tapes over 1 to 20 drives, providing from 50 GB to 9 TB of storage.

With compression, DLT Type IV format supports up to 70 GB capacity.

Data is recorded onto the tape in tracks parallel to the direction of travel (just like QIC tapes). Two tracks are written at once. Read/write head lifetimes are relatively long; once the tape stops moving, there is no relative motion between the heads and the tape.

17.9.5. AIT

AIT is a new format from Sony, and can hold up to 50 GB (with compression) per tape. The tapes contain memory chips which retain an index of the tape's contents. This index can be rapidly read by the tape drive to determine the position of files on the tape, instead of the several minutes that would be required for other tapes. Software such as **SAMS:Alexandria** can operate forty or more AIT tape libraries, communicating directly with the tape's memory chip to display the contents on screen, determine what files were backed up to which tape, locate the correct tape, load it, and restore the data from the tape.

Libraries like this cost in the region of \$20,000, pricing them a little out of the hobbyist market.

17.9.6. Using a New Tape for the First Time

The first time that you try to read or write a new, completely blank tape, the operation will fail. The console messages should be similar to:

```
sa0(ncr1:4:0): NOT READY asc:4,1
sa0(ncr1:4:0): Logical unit is in process of becoming ready
```

The tape does not contain an Identifier Block (block number 0). All QIC tape drives since the adoption of QIC-525 standard write an Identifier Block to the tape. There are two solutions:

- `mt fsf 1` causes the tape drive to write an Identifier Block to the tape.
- Use the front panel button to eject the tape.

Re-insert the tape and dump data to the tape.

dump will report DUMP: End of tape detected and the console will show: `HARDWARE FAILURE info:280 asc:80,96`.

rewind the tape using: `mt rewind`.

Subsequent tape operations are successful.

17.10. Backups to Floppies

17.10.1. Can I Use Floppies for Backing Up My Data?

Floppy disks are not really a suitable media for making backups as:

- The media is unreliable, especially over long periods of time.
- Backing up and restoring is very slow.

- They have a very limited capacity (the days of backing up an entire hard disk onto a dozen or so floppies has long since passed).

However, if you have no other method of backing up your data then floppy disks are better than no backup at all.

If you do have to use floppy disks then ensure that you use good quality ones. Floppies that have been lying around the office for a couple of years are a bad choice. Ideally use new ones from a reputable manufacturer.

17.10.2. So How Do I Backup My Data to Floppies?

The best way to backup to floppy disk is to use `tar(1)` with the `-M` (multi volume) option, which allows backups to span multiple floppies.

To backup all the files in the current directory and sub-directory use this (as `root`):

```
# tar Mcvf /dev/fd0 *
```

When the first floppy is full `tar(1)` will prompt you to insert the next volume (because `tar(1)` is media independent it refers to volumes; in this context it means floppy disk).

Prepare volume #2 for `/dev/fd0` and hit return:

This is repeated (with the volume number incrementing) until all the specified files have been archived.

17.10.3. Can I Compress My Backups?

Unfortunately, `tar(1)` will not allow the `-z` option to be used for multi-volume archives. You could, of course, `gzip(1)` all the files, `tar(1)` them to the floppies, then `gunzip(1)` the files again!

17.10.4. How Do I Restore My Backups?

To restore the entire archive use:

```
# tar Mxvf /dev/fd0
```

There are two ways that you can use to restore only specific files. First, you can start with the first floppy and use:

```
# tar Mxvf /dev/fd0 filename
```

The utility `tar(1)` will prompt you to insert subsequent floppies until it finds the required file.

Alternatively, if you know which floppy the file is on then you can simply insert that floppy and use the same command as above. Note that if the first file on the floppy is a continuation from the previous one then `tar(1)` will warn you that it cannot restore it, even if you have not asked it to!

17.11. Backup Strategies

Original work by Lowell Gilbert.

The first requirement in devising a backup plan is to make sure that all of the following problems are covered:

- Disk failure
- Accidental file deletion
- Random file corruption
- Complete machine destruction (e.g. fire), including destruction of any on-site backups.

It is perfectly possible that some systems will be best served by having each of these problems covered by a completely different technique. Except for strictly personal systems with very low-value data, it is unlikely that one technique would cover all of them.

Some of the techniques in the toolbox are:

- Archives of the whole system, backed up onto permanent media offsite. This actually provides protection against all of the possible problems listed above, but is slow and inconvenient to restore from. You can keep copies of the backups onsite and/or online, but there will still be inconveniences in restoring files, especially for non-privileged users.
- Filesystem snapshots. This is really only helpful in the accidental file deletion scenario, but it can be *very* helpful in that case, and is quick and easy to deal with.
- Copies of whole filesystems and/or disks (e.g. periodic rsync of the whole machine). This is generally most useful in networks with unique requirements. For general protection against disk failure, it is usually inferior to RAID. For restoring accidentally deleted files, it can be comparable to UFS snapshots, but that depends on your preferences.
- RAID. Minimizes or avoids downtime when a disk fails. At the expense of having to deal with disk failures more often (because you have more disks), albeit at a much lower urgency.
- Checking fingerprints of files. The mtree(8) utility is very useful for this. Although it is not a backup technique, it helps guarantee that you will notice when you need to resort to your backups. This is particularly important for offline backups, and should be checked periodically.

It is quite easy to come up with even more techniques, many of them variations on the ones listed above. Specialized requirements will usually lead to specialized techniques (for example, backing up a live database usually requires a method particular to the database software as an intermediate step). The important thing is to know what dangers you want to protect against, and how you will handle each.

17.12. Backup Basics

The three major backup programs are `dump(8)`, `tar(1)`, and `cpio(1)`.

17.12.1. Dump and Restore

The traditional UNIX backup programs are `dump` and `restore`. They operate on the drive as a collection of disk blocks, below the abstractions of files, links and directories that are created by the file systems. `dump` backs up an

entire file system on a device. It is unable to backup only part of a file system or a directory tree that spans more than one file system. `dump` does not write files and directories to tape, but rather writes the raw data blocks that comprise files and directories.

Notatka: If you use `dump` on your root directory, you would not back up `/home`, `/usr` or many other directories since these are typically mount points for other file systems or symbolic links into those file systems.

`dump` has quirks that remain from its early days in Version 6 of AT&T UNIX (circa 1975). The default parameters are suitable for 9-track tapes (6250 bpi), not the high-density media available today (up to 62,182 fpi). These defaults must be overridden on the command line to utilize the capacity of current tape drives.

It is also possible to backup data across the network to a tape drive attached to another computer with `rdump` and `rrestore`. Both programs rely upon `rcmd(3)` and `ruserok(3)` to access the remote tape drive. Therefore, the user performing the backup must be listed in the `.rhosts` file on the remote computer. The arguments to `rdump` and `rrestore` must be suitable to use on the remote computer. When `rdumping` from a FreeBSD computer to an Exabyte tape drive connected to a Sun called `komodo`, use:

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

Beware: there are security implications to allowing `.rhosts` authentication. Evaluate your situation carefully.

It is also possible to use `dump` and `restore` in a more secure fashion over `ssh`.

Przykład 17-1. Using `dump` over `ssh`

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish \
    targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-10.gz
```

Or using `dump`'s built-in method, setting the environment variable `RSH`:

Przykład 17-2. Using `dump` over `ssh` with `RSH` set

```
# RSH=/usr/bin/ssh /sbin/dump -0uan -f targetuser@targetmachine.example.com:/dev/sa0 /usr
```

17.12.2. `tar`

`tar(1)` also dates back to Version 6 of AT&T UNIX (circa 1975). `tar` operates in cooperation with the file system; it writes files and directories to tape. `tar` does not support the full range of options that are available from `cpio(1)`, but it does not require the unusual command pipeline that `cpio` uses.

On FreeBSD 5.3 and later, both GNU `tar` and the default `bsdtar` are available. The GNU version can be invoked with `gtar`. It supports remote devices using the same syntax as `rdump`. To `tar` to an Exabyte tape drive connected to a Sun called `komodo`, use:

```
# /usr/bin/gtar cf komodo:/dev/nsa8 . 2>&1
```

The same could be accomplished with `bsdtar` by using a pipeline and `rsh` to send the data to a remote tape drive.

```
# tar cf - . | rsh hostname dd of=tape-device obs=20b
```

If you are worried about the security of backing up over a network you should use the `ssh` command instead of `rsh`.

17.12.3. `cpio`

`cpio(1)` is the original UNIX file interchange tape program for magnetic media. `cpio` has options (among many others) to perform byte-swapping, write a number of different archive formats, and pipe the data to other programs. This last feature makes `cpio` an excellent choice for installation media. `cpio` does not know how to walk the directory tree and a list of files must be provided through `stdin`.

`cpio` does not support backups across the network. You can use a pipeline and `rsh` to send the data to a remote tape drive.

```
# for f in directory_list; do
find $f >> backup.list
done
# cpio -v -o --format=newc < backup.list | ssh user@host "cat > backup_device"
```

Where `directory_list` is the list of directories you want to back up, `user@host` is the user/hostname combination that will be performing the backups, and `backup_device` is where the backups should be written to (e.g., `/dev/nsa0`).

17.12.4. `pax`

`pax(1)` is IEEE/POSIX's answer to `tar` and `cpio`. Over the years the various versions of `tar` and `cpio` have gotten slightly incompatible. So rather than fight it out to fully standardize them, POSIX created a new archive utility. `pax` attempts to read and write many of the various `cpio` and `tar` formats, plus new formats of its own. Its command set more resembles `cpio` than `tar`.

17.12.5. **Amanda**

Amanda (Advanced Maryland Network Disk Archiver) is a client/server backup system, rather than a single program. An **Amanda** server will backup to a single tape drive any number of computers that have **Amanda** clients and a network connection to the **Amanda** server. A common problem at sites with a number of large disks is that the length of time required to backup to data directly to tape exceeds the amount of time available for the task. **Amanda** solves this problem. **Amanda** can use a „holding disk” to backup several file systems at the same time. **Amanda** creates „archive sets”: a group of tapes used over a period of time to create full backups of all the file systems listed in **Amanda**'s configuration file. The „archive set” also contains nightly incremental (or differential) backups of all the file systems. Restoring a damaged file system requires the most recent full backup and the incremental backups.

The configuration file provides fine control of backups and the network traffic that **Amanda** generates. **Amanda** will use any of the above backup programs to write the data to tape. **Amanda** is available as either a port or a package, it is not installed by default.

17.12.6. Do Nothing

„Do nothing” is not a computer program, but it is the most widely used backup strategy. There are no initial costs. There is no backup schedule to follow. Just say no. If something happens to your data, grin and bear it!

If your time and your data is worth little to nothing, then „Do nothing” is the most suitable backup program for your computer. But beware, UNIX is a useful tool, you may find that within six months you have a collection of files that are valuable to you.

„Do nothing” is the correct backup method for `/usr/obj` and other directory trees that can be exactly recreated by your computer. An example is the files that comprise the HTML or PostScript version of this Handbook. These document formats have been created from SGML input files. Creating backups of the HTML or PostScript files is not necessary. The SGML files are backed up regularly.

17.12.7. Which Backup Program Is Best?

`dump(8)` *Period*. Elizabeth D. Zwicky torture tested all the backup programs discussed here. The clear choice for preserving all your data and all the peculiarities of UNIX file systems is `dump`. Elizabeth created file systems containing a large variety of unusual conditions (and some not so unusual ones) and tested each program by doing a backup and restore of those file systems. The peculiarities included: files with holes, files with holes and a block of nulls, files with funny characters in their names, unreadable and unwritable files, devices, files that change size during the backup, files that are created/deleted during the backup and more. She presented the results at LISA V in Oct. 1991. See torture-testing Backup and Archive Programs (<http://berdmann.dyndns.org/zwicky/testdump.doc.html>).

17.12.8. Emergency Restore Procedure

17.12.8.1. Before the Disaster

There are only four steps that you need to perform in preparation for any disaster that may occur.

First, print the `bsdlabeled` from each of your disks (e.g. `bsdlabeled da0 | lpr`), your file system table (`/etc/fstab`) and all boot messages, two copies of each.

Second, determine that the boot and fix-it floppies (`boot.flp` and `fixit.flp`) have all your devices. The easiest way to check is to reboot your machine with the boot floppy in the floppy drive and check the boot messages. If all your devices are listed and functional, skip on to step three.

Otherwise, you have to create two custom bootable floppies which have a kernel that can mount all of your disks and access your tape drive. These floppies must contain: `fdisk`, `bsdlabeled`, `newfs`, `mount`, and whichever backup program you use. These programs must be statically linked. If you use `dump`, the floppy must contain `restore`.

Third, create backup tapes regularly. Any changes that you make after your last backup may be irretrievably lost. Write-protect the backup tapes.

Fourth, test the floppies (either `boot.flp` and `fixit.flp` or the two custom bootable floppies you made in step two.) and backup tapes. Make notes of the procedure. Store these notes with the bootable floppy, the printouts and the backup tapes. You will be so distraught when restoring that the notes may prevent you from destroying your backup tapes (How? In place of `tar xvf /dev/sa0`, you might accidentally type `tar cvf /dev/sa0` and over-write your backup tape).

For an added measure of security, make bootable floppies and two backup tapes each time. Store one of each at a remote location. A remote location is NOT the basement of the same office building. A number of firms in the World Trade Center learned this lesson the hard way. A remote location should be physically separated from your computers and disk drives by a significant distance.

Przykład 17-3. A Script for Creating a Bootable Floppy

```
#!/bin/sh
#
# create a restore floppy
#
# format the floppy
#
PATH=/bin:/sbin:/usr/sbin:/usr/bin

fdformat -q fd0
if [ $? -ne 0 ]
then
    echo "Bad floppy, please use a new one"
    exit 1
fi

# place boot blocks on the floppy
#
bsdlabel -w -B /dev/fd0c fd1440

#
# newfs the one and only partition
#
newfs -t 2 -u 18 -l 1 -c 40 -i 5120 -m 5 -o space /dev/fd0a

#
# mount the new floppy
#
mount /dev/fd0a /mnt

#
# create required directories
#
mkdir /mnt/dev
mkdir /mnt/bin
mkdir /mnt/sbin
mkdir /mnt/etc
mkdir /mnt/root
mkdir /mnt/mnt                # for the root partition
mkdir /mnt/tmp
mkdir /mnt/var

#
# populate the directories
#
if [ ! -x /sys/compile/MINI/kernel ]
then
    cat << EOM
The MINI kernel does not exist, please create one.
Here is an example config file:
#
# MINI -- A kernel to get FreeBSD onto a disk.
```

```

#
machine      "i386"
cpu          "I486_CPU"
ident        MINI
maxusers     5

options      INET                # needed for _tcp _icmpstat _ipstat
                                #           _udpstat _tcpstat _udb
options      FFS                 #Berkeley Fast File System
options      FAT_CURSOR          #block cursor in syscons or pccons
options      SCSI_DELAY=15       #Be pessimistic about Joe SCSI device
options      NCONS=2             #1 virtual consoles
options      USERCONFIG         #Allow user configuration with -c XXX

config       kernel  root on da0 swap on da0 and da1 dumps on da0

device       isa0
device       pci0

device       fdc0      at isa? port "IO_FD1" bio irq 6 drq 2 vector fdintr
device       fd0       at fdc0 drive 0

device       ncr0

device       scbus0

device       sc0       at isa? port "IO_KBD" tty irq 1 vector scintr
device       npx0      at isa? port "IO_NPX" irq 13 vector npxintr

device       da0
device       da1
device       da2

device       sa0

pseudo-device loop        # required by INET
pseudo-device gzip        # Exec gzipped a.out's
EOM

        exit 1
fi

cp -f /sys/compile/MINI/kernel /mnt

gzip -c -best /sbin/init > /mnt/sbin/init
gzip -c -best /sbin/fsck > /mnt/sbin/fsck
gzip -c -best /sbin/mount > /mnt/sbin/mount
gzip -c -best /sbin/halt > /mnt/sbin/halt
gzip -c -best /sbin/restore > /mnt/sbin/restore

gzip -c -best /bin/sh > /mnt/bin/sh
gzip -c -best /bin/sync > /mnt/bin/sync

cp /root/.profile /mnt/root

```

```

cp -f /dev/MAKEDEV /mnt/dev
chmod 755 /mnt/dev/MAKEDEV

chmod 500 /mnt/sbin/init
chmod 555 /mnt/sbin/fsck /mnt/sbin/mount /mnt/sbin/halt
chmod 555 /mnt/bin/sh /mnt/bin/sync
chmod 6555 /mnt/sbin/restore

#
# create the devices nodes
#
cd /mnt/dev
./MAKEDEV std
./MAKEDEV da0
./MAKEDEV da1
./MAKEDEV da2
./MAKEDEV sa0
./MAKEDEV pty0
cd /

#
# create minimum file system table
#
cat > /mnt/etc/fstab <<EOM
/dev/fd0a    /      ufs    rw  1  1
EOM

#
# create minimum passwd file
#
cat > /mnt/etc/passwd <<EOM
root:*:0:0:Charlie &:/root:/bin/sh
EOM

cat > /mnt/etc/master.passwd <<EOM
root::0:0::0:0:Charlie &:/root:/bin/sh
EOM

chmod 600 /mnt/etc/master.passwd
chmod 644 /mnt/etc/passwd
/usr/sbin/pwd_mkdb -d/mnt/etc /mnt/etc/master.passwd

#
# umount the floppy and inform the user
#
/sbin/umount /mnt
echo "The floppy has been unmounted and is now ready."

```

17.12.8.2. After the Disaster

The key question is: did your hardware survive? You have been doing regular backups so there is no need to worry about the software.

If the hardware has been damaged, the parts should be replaced before attempting to use the computer.

If your hardware is okay, check your floppies. If you are using a custom boot floppy, boot single-user (type `-s` at the `boot:` prompt). Skip the following paragraph.

If you are using the `boot.flp` and `fixit.flp` floppies, keep reading. Insert the `boot.flp` floppy in the first floppy drive and boot the computer. The original install menu will be displayed on the screen. Select the `Fixit--Repair` mode with `CDROM` or `floppy.` option. Insert the `fixit.flp` when prompted. `restore` and the other programs that you need are located in `/mnt2/rescue` (`/mnt2/stand` for FreeBSD versions older than 5.2).

Recover each file system separately.

Try to mount (e.g. `mount /dev/da0a /mnt`) the root partition of your first disk. If the `bsdlabel` was damaged, use `bsdlabel` to re-partition and label the disk to match the label that you printed and saved. Use `newfs` to re-create the file systems. Re-mount the root partition of the floppy read-write (`mount -u -o rw /mnt`). Use your backup program and backup tapes to recover the data for this file system (e.g. `restore vrf /dev/sa0`). Unmount the file system (e.g. `umount /mnt`). Repeat for each file system that was damaged.

Once your system is running, backup your data onto new tapes. Whatever caused the crash or data loss may strike again. Another hour spent now may save you from further distress later.

17.13. Network, Memory, and File-Backed File Systems

Reorganized and enhanced by Marc Fonvieille.

Aside from the disks you physically insert into your computer: floppies, CDs, hard drives, and so forth; other forms of disks are understood by FreeBSD - the *virtual disks*.

These include network file systems such as the Network File System and Coda, memory-based file systems and file-backed file systems.

According to the FreeBSD version you run, you will have to use different tools for creation and use of file-backed and memory-based file systems.

Notatka: Use `devfs(5)` to allocate device nodes transparently for the user.

17.13.1. File-Backed File System

The utility `mdconfig(8)` is used to configure and enable memory disks, `md(4)`, under FreeBSD. To use `mdconfig(8)`, you have to load `md(4)` module or to add the support in your kernel configuration file:

```
device md
```

The `mdconfig(8)` command supports three kinds of memory backed virtual disks: memory disks allocated with `malloc(9)`, memory disks using a file or swap space as backing. One possible use is the mounting of floppy or CD images kept in files.

To mount an existing file system image:

Przykład 17-4. Using `mdconfig` to Mount an Existing File System Image

```
# mdconfig -a -t vnode -f diskimage -u 0
# mount /dev/md0 /mnt
```

To create a new file system image with `mdconfig(8)`:

Przykład 17-5. Creating a New File-Backed Disk with `mdconfig`

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdconfig -a -t vnode -f newimage -u 0
# bsdlabeled -w md0 auto
# newfs md0a
/dev/md0a: 5.0MB (10224 sectors) block size 16384, fragment size 2048
      using 4 cylinder groups of 1.25MB, 80 blks, 192 inodes.
super-block backups (for fsck -b #) at:
      160, 2720, 5280, 7840
# mount /dev/md0a /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md0a      4710    4  4330    0%    /mnt
```

If you do not specify the unit number with the `-u` option, `mdconfig(8)` will use the `md(4)` automatic allocation to select an unused device. The name of the allocated unit will be output on stdout like `md4`. For more details about `mdconfig(8)`, please refer to the manual page.

The utility `mdconfig(8)` is very useful, however it asks many command lines to create a file-backed file system. FreeBSD also comes with a tool called `mdmfs(8)`, this program configures a `md(4)` disk using `mdconfig(8)`, puts a UFS file system on it using `newfs(8)`, and mounts it using `mount(8)`. For example, if you want to create and mount the same file system image as above, simply type the following:

Przykład 17-6. Configure and Mount a File-Backed Disk with `mdmfs`

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdmfs -F newimage -s 5m md0 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md0      4718    4  4338    0%    /mnt
```

If you use the option `md` without unit number, `mdmfs(8)` will use `md(4)` auto-unit feature to automatically select an unused device. For more details about `mdmfs(8)`, please refer to the manual page.

17.13.2. Memory-Based File System

For a memory-based file system the „swap backing” should normally be used. Using swap backing does not mean that the memory disk will be swapped out to disk by default, but merely that the memory disk will be allocated from a memory pool which can be swapped out to disk if needed. It is also possible to create memory-based disk which are malloc(9) backed, but using malloc backed memory disks, especially large ones, can result in a system panic if the kernel runs out of memory.

Przykład 17-7. Creating a New Memory-Based Disk with `mdconfig`

```
# mdconfig -a -t swap -s 5m -u 1
# newfs -U md1
/dev/md1: 5.0MB (10240 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.27MB, 81 blks, 192 inodes.
        with soft updates
super-block backups (for fsck -b #) at:
 160, 2752, 5344, 7936
# mount /dev/md1 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md1      4718    4 4338    0%    /mnt
```

Przykład 17-8. Creating a New Memory-Based Disk with `mdmfs`

```
# mdmfs -s 5m md2 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md2      4846    2 4458    0%    /mnt
```

17.13.3. Detaching a Memory Disk from the System

When a memory-based or file-based file system is not used, you should release all resources to the system. The first thing to do is to unmount the file system, then use `mdconfig(8)` to detach the disk from the system and release the resources.

For example to detach and free all resources used by `/dev/md4`:

```
# mdconfig -d -u 4
```

It is possible to list information about configured `md(4)` devices in using the command `mdconfig -l`.

17.14. File System Snapshots

Contributed by Tom Rhodes.

FreeBSD offers a feature in conjunction with Soft Updates: File system snapshots.

Snapshots allow a user to create images of specified file systems, and treat them as a file. Snapshot files must be created in the file system that the action is performed on, and a user may create no more than 20 snapshots per file

system. Active snapshots are recorded in the superblock so they are persistent across unmount and remount operations along with system reboots. When a snapshot is no longer required, it can be removed with the standard `rm(1)` command. Snapshots may be removed in any order, however all the used space may not be acquired because another snapshot will possibly claim some of the released blocks.

The un-alterable `snapshot` file flag is set by `mksnap_ffs(8)` after initial creation of a snapshot file. The `unlink(1)` command makes an exception for snapshot files since it allows them to be removed.

Snapshots are created with the `mount(8)` command. To place a snapshot of `/var` in the file `/var/snapshot/snap` use the following command:

```
# mount -u -o snapshot /var/snapshot/snap /var
```

Alternatively, you can use `mksnap_ffs(8)` to create a snapshot:

```
# mksnap_ffs /var /var/snapshot/snap
```

One can find snapshot files on a file system (e.g. `/var`) by using the `find(1)` command:

```
# find /var -flags snapshot
```

Once a snapshot has been created, it has several uses:

- Some administrators will use a snapshot file for backup purposes, because the snapshot can be transfered to CDs or tape.
- The file system integrity checker, `fsck(8)`, may be run on the snapshot. Assuming that the file system was clean when it was mounted, you should always get a clean (and unchanging) result. This is essentially what the background `fsck(8)` process does.
- Run the `dump(8)` utility on the snapshot. A dump will be returned that is consistent with the file system and the timestamp of the snapshot. `dump(8)` can also take a snapshot, create a dump image and then remove the snapshot in one command using the `-L` flag.
- `mount(8)` the snapshot as a frozen image of the file system. To `mount(8)` the snapshot `/var/snapshot/snap` run:

```
# mdconfig -a -t vnode -f /var/snapshot/snap -u 4
# mount -r /dev/md4 /mnt
```

You can now walk the hierarchy of your frozen `/var` file system mounted at `/mnt`. Everything will initially be in the same state it was during the snapshot creation time. The only exception is that any earlier snapshots will appear as zero length files. When the use of a snapshot has delimited, it can be unmounted with:

```
# umount /mnt
# mdconfig -d -u 4
```

For more information about `softupdates` and file system snapshots, including technical papers, you can visit Marshall Kirk McKusick's website at <http://www.mckusick.com/>.

17.15. File System Quotas

Quotas are an optional feature of the operating system that allow you to limit the amount of disk space and/or the number of files a user or members of a group may allocate on a per-file system basis. This is used most often on

timesharing systems where it is desirable to limit the amount of resources any one user or group of users may allocate. This will prevent one user or group of users from consuming all of the available disk space.

17.15.1. Configuring Your System to Enable Disk Quotas

Before attempting to use disk quotas, it is necessary to make sure that quotas are configured in your kernel. This is done by adding the following line to your kernel configuration file:

```
options QUOTA
```

The stock `GENERIC` kernel does not have this enabled by default, so you will have to configure, build and install a custom kernel in order to use disk quotas. Please refer to Rozdział 8 for more information on kernel configuration.

Next you will need to enable disk quotas in `/etc/rc.conf`. This is done by adding the line:

```
enable_quotas="YES"
```

For finer control over your quota startup, there is an additional configuration variable available. Normally on bootup, the quota integrity of each file system is checked by the `quotacheck(8)` program. The `quotacheck(8)` facility insures that the data in the quota database properly reflects the data on the file system. This is a very time consuming process that will significantly affect the time your system takes to boot. If you would like to skip this step, a variable in `/etc/rc.conf` is made available for the purpose:

```
check_quotas="NO"
```

Finally you will need to edit `/etc/fstab` to enable disk quotas on a per-file system basis. This is where you can either enable user or group quotas or both for all of your file systems.

To enable per-user quotas on a file system, add the `userquota` option to the options field in the `/etc/fstab` entry for the file system you want to enable quotas on. For example:

```
/dev/dals2g    /home    ufs rw,userquota 1 2
```

Similarly, to enable group quotas, use the `groupquota` option instead of `userquota`. To enable both user and group quotas, change the entry as follows:

```
/dev/dals2g    /home    ufs rw,userquota,groupquota 1 2
```

By default, the quota files are stored in the root directory of the file system with the names `quota.user` and `quota.group` for user and group quotas respectively. See `fstab(5)` for more information. Even though the `fstab(5)` manual page says that you can specify an alternate location for the quota files, this is not recommended because the various quota utilities do not seem to handle this properly.

At this point you should reboot your system with your new kernel. `/etc/rc` will automatically run the appropriate commands to create the initial quota files for all of the quotas you enabled in `/etc/fstab`, so there is no need to manually create any zero length quota files.

In the normal course of operations you should not be required to run the `quotacheck(8)`, `quotaon(8)`, or `quotaoff(8)` commands manually. However, you may want to read their manual pages just to be familiar with their operation.

17.15.2. Setting Quota Limits

Once you have configured your system to enable quotas, verify that they really are enabled. An easy way to do this is to run:

```
# quota -v
```

You should see a one line summary of disk usage and current quota limits for each file system that quotas are enabled on.

You are now ready to start assigning quota limits with the `edquota(8)` command.

You have several options on how to enforce limits on the amount of disk space a user or group may allocate, and how many files they may create. You may limit allocations based on disk space (block quotas) or number of files (inode quotas) or a combination of both. Each of these limits are further broken down into two categories: hard and soft limits.

A hard limit may not be exceeded. Once a user reaches his hard limit he may not make any further allocations on the file system in question. For example, if the user has a hard limit of 500 kbytes on a file system and is currently using 490 kbytes, the user can only allocate an additional 10 kbytes. Attempting to allocate an additional 11 kbytes will fail.

Soft limits, on the other hand, can be exceeded for a limited amount of time. This period of time is known as the grace period, which is one week by default. If a user stays over his or her soft limit longer than the grace period, the soft limit will turn into a hard limit and no further allocations will be allowed. When the user drops back below the soft limit, the grace period will be reset.

The following is an example of what you might see when you run the `edquota(8)` command. When the `edquota(8)` command is invoked, you are placed into the editor specified by the `EDITOR` environment variable, or in the `vi` editor if the `EDITOR` variable is not set, to allow you to edit the quota limits.

```
# edquota -u test
```

```
Quotas for user test:
```

```
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
      inodes in use: 7, limits (soft = 50, hard = 60)
/usr/var: kbytes in use: 0, limits (soft = 50, hard = 75)
          inodes in use: 0, limits (soft = 50, hard = 60)
```

You will normally see two lines for each file system that has quotas enabled. One line for the block limits, and one line for inode limits. Simply change the value you want updated to modify the quota limit. For example, to raise this user's block limit from a soft limit of 50 and a hard limit of 75 to a soft limit of 500 and a hard limit of 600, change:

```
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
```

to:

```
/usr: kbytes in use: 65, limits (soft = 500, hard = 600)
```

The new quota limits will be in place when you exit the editor.

Sometimes it is desirable to set quota limits on a range of UIDs. This can be done by use of the `-p` option on the `edquota(8)` command. First, assign the desired quota limit to a user, and then run `edquota -p protouser startuid-enduid`. For example, if user `test` has the desired quota limits, the following command can be used to duplicate those quota limits for UIDs 10,000 through 19,999:

```
# edquota -p test 10000-19999
```

For more information see `edquota(8)` manual page.

17.15.3. Checking Quota Limits and Disk Usage

You can use either the `quota(1)` or the `repquota(8)` commands to check quota limits and disk usage. The `quota(1)` command can be used to check individual user or group quotas and disk usage. A user may only examine his own quota, and the quota of a group he is a member of. Only the super-user may view all user and group quotas. The `repquota(8)` command can be used to get a summary of all quotas and disk usage for file systems with quotas enabled.

The following is some sample output from the `quota -v` command for a user that has quota limits on two file systems.

```
Disk quotas for user test (uid 1002):
  Filesystem  usage    quota   limit   grace   files   quota   limit   grace
    /usr      65*      50      75     5days      7      50      60
  /usr/var      0       50      75              0      50      60
```

On the `/usr` file system in the above example, this user is currently 15 kbytes over the soft limit of 50 kbytes and has 5 days of the grace period left. Note the asterisk `*` which indicates that the user is currently over his quota limit.

Normally file systems that the user is not using any disk space on will not show up in the output from the `quota(1)` command, even if he has a quota limit assigned for that file system. The `-v` option will display those file systems, such as the `/usr/var` file system in the above example.

17.15.4. Quotas over NFS

Quotas are enforced by the quota subsystem on the NFS server. The `rpc.rquotad(8)` daemon makes quota information available to the `quota(1)` command on NFS clients, allowing users on those machines to see their quota statistics.

Enable `rpc.rquotad` in `/etc/inetd.conf` like so:

```
rquotad/1      dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

Now restart `inetd`:

```
# kill -HUP `cat /var/run/inetd.pid`
```

17.16. Encrypting Disk Partitions

Contributed by Lucky Green.

FreeBSD offers excellent online protections against unauthorized data access. File permissions and Mandatory Access Control (MAC) (see Rozdział 15) help prevent unauthorized third-parties from accessing data while the operating system is active and the computer is powered up. However, the permissions enforced by the operating system are irrelevant if an attacker has physical access to a computer and can simply move the computer's hard drive to another system to copy and analyze the sensitive data.

Regardless of how an attacker may have come into possession of a hard drive or powered-down computer, both **GEOM Based Disk Encryption (gbde)** and `geli` cryptographic subsystems in FreeBSD are able to protect the data on the computer's file systems against even highly-motivated attackers with significant resources. Unlike cumbersome encryption methods that encrypt only individual files, `gbde` and `geli` transparently encrypt entire file systems. No cleartext ever touches the hard drive's platter.

17.16.1. Disk Encryption with gbde

1. Become root

Configuring **gbde** requires super-user privileges.

```
% su -
Password:
```

2. Add `gbde(4)` Support to the Kernel Configuration File

Add the following line to the kernel configuration file:

```
options GEOM_BDE
```

Rebuild the kernel as described in Rozdział 8.

Reboot into the new kernel.

17.16.1.1. Preparing the Encrypted Hard Drive

The following example assumes that you are adding a new hard drive to your system that will hold a single encrypted partition. This partition will be mounted as `/private`. **gbde** can also be used to encrypt `/home` and `/var/mail`, but this requires more complex instructions which exceed the scope of this introduction.

1. Add the New Hard Drive

Install the new drive to the system as explained in Sekcja 17.3. For the purposes of this example, a new hard drive partition has been added as `/dev/ad4s1c`. The `/dev/ad0s1*` devices represent existing standard FreeBSD partitions on the example system.

```
# ls /dev/ad*
/dev/ad0          /dev/ad0s1b      /dev/ad0s1e      /dev/ad4s1
/dev/ad0s1        /dev/ad0s1c      /dev/ad0s1f      /dev/ad4s1c
/dev/ad0s1a       /dev/ad0s1d      /dev/ad4
```

2. Create a Directory to Hold `gbde` Lock Files

```
# mkdir /etc/gbde
```

The **gbde** lock file contains information that **gbde** requires to access encrypted partitions. Without access to the lock file, **gbde** will not be able to decrypt the data contained in the encrypted partition without significant manual intervention which is not supported by the software. Each encrypted partition uses a separate lock file.

3. Initialize the `gbde` Partition

A **gbde** partition must be initialized before it can be used. This initialization needs to be performed only once:

```
# gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c
```

gbde(8) will open your editor, permitting you to set various configuration options in a template. For use with UFS1 or UFS2, set the `sector_size` to 2048:

```
$FreeBSD: src/sbin/gbde/template.txt,v 1.1 2002/10/20 11:16:13 phk Exp $
#
# Sector size is the smallest unit of data which can be read or written.
# Making it too small decreases performance and decreases available space.
# Making it too large may prevent filesystems from working. 512 is the
# minimum and always safe. For UFS, use the fragment size
#
sector_size      =          2048
[...]
```

gbde(8) will ask you twice to type the passphrase that should be used to secure the data. The passphrase must be the same both times. **gbde**'s ability to protect your data depends entirely on the quality of the passphrase that you choose.¹

The `gbde init` command creates a lock file for your **gbde** partition that in this example is stored as `/etc/gbde/ad4s1c`.

Uwaga! **gbde** lock files *must* be backed up together with the contents of any encrypted partitions. While deleting a lock file alone cannot prevent a determined attacker from decrypting a **gbde** partition, without the lock file, the legitimate owner will be unable to access the data on the encrypted partition without a significant amount of work that is totally unsupported by gbde(8) and its designer.

4. Attach the Encrypted Partition to the Kernel

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c
```

You will be asked to provide the passphrase that you selected during the initialization of the encrypted partition. The new encrypted device will show up in `/dev` as `/dev/device_name.bde`:

```
# ls /dev/ad*
/dev/ad0          /dev/ad0s1b      /dev/ad0s1e      /dev/ad4s1
/dev/ad0s1        /dev/ad0s1c      /dev/ad0s1f      /dev/ad4s1c
/dev/ad0s1a       /dev/ad0s1d      /dev/ad4         /dev/ad4s1c.bde
```

5. Create a File System on the Encrypted Device

Once the encrypted device has been attached to the kernel, you can create a file system on the device. To create a file system on the encrypted device, use `newfs(8)`. Since it is much faster to initialize a new UFS2 file system than it is to initialize the old UFS1 file system, using `newfs(8)` with the `-O2` option is recommended.

```
# newfs -U -O2 /dev/ad4s1c.bde
```

Notatka: The `newfs(8)` command must be performed on an attached **gbde** partition which is identified by a `*.bde` extension to the device name.

6. Mount the Encrypted Partition

Create a mount point for the encrypted file system.

```
# mkdir /private
```

Mount the encrypted file system.

```
# mount /dev/ad4s1c.bde /private
```

7. Verify That the Encrypted File System is Available

The encrypted file system should now be visible to `df(1)` and be available for use.

```
% df -H
Filesystem      Size   Used  Avail Capacity  Mounted on
/dev/ad0s1a     1037M    72M   883M      8%      /
/devfs          1.0K    1.0K     0B    100%    /dev
/dev/ad0s1f      8.1G    55K    7.5G      0%    /home
/dev/ad0s1e     1037M    1.1M   953M      0%    /tmp
/dev/ad0s1d      6.1G    1.9G    3.7G     35%    /usr
/dev/ad4s1c.bde  150G    4.1K   138G      0%    /private
```

17.16.1.2. Mounting Existing Encrypted File Systems

After each boot, any encrypted file systems must be re-attached to the kernel, checked for errors, and mounted, before the file systems can be used. The required commands must be executed as user `root`.

1. Attach the gbde Partition to the Kernel

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c
```

You will be asked to provide the passphrase that you selected during initialization of the encrypted **gbde** partition.

2. Check the File System for Errors

Since encrypted file systems cannot yet be listed in `/etc/fstab` for automatic mounting, the file systems must be checked for errors by running `fsck(8)` manually before mounting.

```
# fsck -p -t ffs /dev/ad4s1c.bde
```

3. Mount the Encrypted File System

```
# mount /dev/ad4s1c.bde /private
```

The encrypted file system is now available for use.

17.16.1.2.1. Automatically Mounting Encrypted Partitions

It is possible to create a script to automatically attach, check, and mount an encrypted partition, but for security reasons the script should not contain the `gbde(8)` password. Instead, it is recommended that such scripts be run manually while providing the password via the console or `ssh(1)`.

As of FreeBSD 5.2-RELEASE, there is a new `rc.d` script provided. Arguments for this script can be passed via `rc.conf(5)`, for example:

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
```

This will require that the **gbde** passphrase be entered at boot time. After typing the correct passphrase, the **gbde** encrypted partition will be mounted automatically. This can be very useful when using **gbde** on notebooks.

17.16.1.3. Cryptographic Protections Employed by gbde

gbde(8) encrypts the sector payload using 128-bit AES in CBC mode. Each sector on the disk is encrypted with a different AES key. For more information on **gbde**'s cryptographic design, including how the sector keys are derived from the user-supplied passphrase, see gbde(4).

17.16.1.4. Compatibility Issues

sysinstall(8) is incompatible with **gbde**-encrypted devices. All *.bde devices must be detached from the kernel before starting sysinstall(8) or it will crash during its initial probing for devices. To detach the encrypted device used in our example, use the following command:

```
# gbde detach /dev/ad4s1c
```

Also note that, as vinum(4) does not use the geom(4) subsystem, you cannot use **gbde** with **vinum** volumes.

17.16.2. Disk Encryption with geli

Contributed by Daniel Gerzo.

A new cryptographic GEOM class is available as of FreeBSD 6.0 - **geli**. It is currently being developed by Pawel Jakub Dawidek <pjd@FreeBSD.org>. Geli is different to gbde; it offers different features and uses a different scheme for doing cryptographic work.

The most important features of geli(8) are:

- Utilizes the crypto(9) framework — when cryptographic hardware is available, geli will use it automatically.
- Supports multiple cryptographic algorithms (currently AES, Blowfish, and 3DES).
- Allows the root partition to be encrypted. The passphrase used to access the encrypted root partition will be requested during the system boot.
- Allows the use of two independent keys (e.g. a „key” and a „company key”).
- geli is fast - performs simple sector-to-sector encryption.
- Allows backup and restore of Master Keys. When a user has to destroy his keys, it will be possible to get access to the data again by restoring keys from the backup.
- Allows to attach a disk with a random, one-time key — useful for swap partitions and temporary file systems.

More geli features can be found in the geli(8) manual page.

The next steps will describe how to enable support for geli in the FreeBSD kernel and will explain how to create a new geli encryption provider. At the end it will be demonstrated how to create an encrypted swap partition using features provided by geli.

In order to use geli, you must be running FreeBSD 6.0-RELEASE or later. Super-user privileges will be required since modifications to the kernel are necessary.

1. Adding geli Support to the Kernel Configuration File

Add the following lines to the kernel configuration file:

```
options GEOM_ELI
device crypto
```

Rebuild the kernel as described in Rozdział 8.

Alternatively, the `geli` module can be loaded at boot time. Add the following line to the `/boot/loader.conf`:

```
geom_eli_load="YES"
```

`geli(8)` should now be supported by the kernel.

2. Generating the Master Key

The following example will describe how to generate a key file, which will be used as part of the Master Key for the encrypted provider mounted under `/private`. The key file will provide some random data used to encrypt the Master Key. The Master Key will be protected by a passphrase as well. Provider's sector size will be 4kB big. Furthermore, the discussion will describe how to attach the `geli` provider, create a file system on it, how to mount it, how to work with it, and finally how to detach it.

It is recommended to use a bigger sector size (like 4kB) for better performance.

The Master Key will be protected with a passphrase and the data source for key file will be `/dev/random`. The sector size of `/dev/da2.eli`, which we call provider, will be 4kB.

```
# dd if=/dev/random of=/root/da2.key bs=64 count=1
# geli init -s 4096 -K /root/da2.key /dev/da2
Enter new passphrase:
Reenter new passphrase:
```

It is not mandatory that both a passphrase and a key file are used; either method of securing the Master Key can be used in isolation.

If key file is given as „-“, standard input will be used. This example shows how more than one key file can be used.

```
# cat keyfile1 keyfile2 keyfile3 | geli init -K - /dev/da2
```

3. Attaching the Provider with the generated Key

```
# geli attach -k /root/da2.key /dev/da2
Enter passphrase:
```

The new plaintext device will be named `/dev/da2.eli`.

```
# ls /dev/da2*
/dev/da2  /dev/da2.eli
```

4. Creating the new File System

```
# dd if=/dev/random of=/dev/da2.eli bs=1m
# newfs /dev/da2.eli
# mount /dev/da2.eli /private
```

The encrypted file system should be visible to `df(1)` and be available for use now.

```
# df -H
```

Filesystem	Size	Used	Avail	Capacity	Mounted on
/dev/ad0s1a	248M	89M	139M	38%	/
/devfs	1.0K	1.0K	0B	100%	/dev
/dev/ad0s1f	7.7G	2.3G	4.9G	32%	/usr
/dev/ad0s1d	989M	1.5M	909M	0%	/tmp
/dev/ad0s1e	3.9G	1.3G	2.3G	35%	/var

```
/dev/da2.eli 150G 4.1K 138G 0% /private
```

5. Unmounting and Detaching the Provider

Once the work on the encrypted partition is done, and the `/private` partition is no longer needed, it is prudent to consider unmounting and detaching the `geli` encrypted partition from the kernel.

```
# umount /private
# geli detach da2.eli
```

More information about the use of `geli(8)` can be found in the manual page.

17.16.2.1. Using the `geli rc.d` Script

`geli` comes with a `rc.d` script which can be used to simplify the usage of `geli`. An example of configuring `geli` through `rc.conf(5)` follows:

```
geli_devices="da2"
geli_da2_flags="-p -k /root/da2.key"
```

This will configure `/dev/da2` as a `geli` provider of which the Master Key file is located in `/root/da2.key`, and `geli` will not use a passphrase when attaching the provider (note that this can only be used if `-P` was given during the `geli` init phase). The system will detach the `geli` provider from the kernel before the system shuts down.

More information about configuring `rc.d` is provided in the `rc.d` section of the Handbook.

17.17. Encrypting Swap Space

Written by Christian Brüffer.

Swap encryption in FreeBSD is easy to configure and has been available since FreeBSD 5.3-RELEASE. Depending on which version of FreeBSD is being used, different options are available and configuration can vary slightly. From FreeBSD 6.0-RELEASE onwards, the `gbde(8)` or `geli(8)` encryption systems can be used for swap encryption. With earlier versions, only `gbde(8)` is available. Both systems use the `encswap rc.d` script.

The previous section, Encrypting Disk Partitions, includes a short discussion on the different encryption systems.

17.17.1. Why should Swap be Encrypted?

Like the encryption of disk partitions, encryption of swap space is done to protect sensitive information. Imagine an application that e.g. deals with passwords. As long as these passwords stay in physical memory, all is well. However, if the operating system starts swapping out memory pages to free space for other applications, the passwords may be written to the disk platters unencrypted and easy to retrieve for an adversary. Encrypting swap space can be a solution for this scenario.

17.17.2. Preparation

Notatka: For the remainder of this section, `ad0s1b` will be the swap partition.

Up to this point the swap has been unencrypted. It is possible that there are already passwords or other sensitive data on the disk platters in cleartext. To rectify this, the data on the swap partition should be overwritten with random garbage:

```
# dd if=/dev/random of=/dev/ad0s1b bs=1m
```

17.17.3. Swap Encryption with gbde(8)

If FreeBSD 6.0-RELEASE or newer is being used, the `.bde` suffix should be added to the device in the respective `/etc/fstab` swap line:

# Device	Mountpoint	FStype	Options	Dump	Pass#
/dev/ad0s1b.bde	none	swap	sw	0	0

For systems prior to FreeBSD 6.0-RELEASE, the following line in `/etc/rc.conf` is also needed:

```
gbde_swap_enable="YES"
```

17.17.4. Swap Encryption with geli(8)

Alternatively, the procedure for using `geli(8)` for swap encryption is similar to that of using `gbde(8)`. The `.eli` suffix should be added to the device in the respective `/etc/fstab` swap line:

# Device	Mountpoint	FStype	Options	Dump	Pass#
/dev/ad0s1b.eli	none	swap	sw	0	0

`geli(8)` uses the AES algorithm with a key length of 256 bit by default.

Optionally, these defaults can be altered using the `geli_swap_flags` option in `/etc/rc.conf`. The following line tells the `encswap rc.d` script to create `geli(8)` swap partitions using the Blowfish algorithm with a key length of 128 bit, a sectorsize of 4 kilobytes and the „detach on last close” option set:

```
geli_swap_flags="-a blowfish -l 128 -s 4096 -d"
```

Please refer to the description of the `onetime` command in the `geli(8)` manual page for a list of possible options.

17.17.5. Verifying that it Works

Once the system has been rebooted, proper operation of the encrypted swap can be verified using the `swapinfo` command.

If `gbde(8)` is being used:

```
% swapinfo
Device          1K-blocks    Used    Avail Capacity
/dev/ad0s1b.bde  542720        0    542720      0%
```

If geli(8) is being used:

```
% swapinfo
Device          1K-blocks    Used    Avail Capacity
/dev/ad0s1b.eli    542720         0    542720     0%
```

Przypisy

1. For tips on how to select a secure passphrase that is easy to remember, see the Diceware Passphrase (<http://world.std.com/~reinhold/diceware.html>) website.

Rozdział 18. GEOM: Modular Disk Transformation Framework

Written by Tom Rhodes.

18.1. Synopsis

This chapter covers the use of disks under the GEOM framework in FreeBSD. This includes the major RAID control utilities which use the framework for configuration. This chapter will not go into in depth discussion on how GEOM handles or controls I/O, the underlying subsystem, or code. This information is provided through the `geom(4)` manual page and its various SEE ALSO references. This chapter is also not a definitive guide to RAID configurations. Only GEOM-supported RAID classifications will be discussed.

After reading this chapter, you will know:

- What type of RAID support is available through GEOM.
- How to use the base utilities to configure, maintain, and manipulate the various RAID levels.
- How to mirror, stripe, encrypt, and remotely connect disk devices through GEOM.
- How to troubleshoot disks attached to the GEOM framework.

Before reading this chapter, you should:

- Understand how FreeBSD treats disk devices (Rozdział 17).
- Know how to configure and install a new FreeBSD kernel (Rozdział 8).

18.2. GEOM Introduction

GEOM permits access and control to classes — Master Boot Records, BSD labels, etc — through the use of providers, or the special files in `/dev`. Supporting various software RAID configurations, GEOM will transparently provide access to the operating system and operating system utilities.

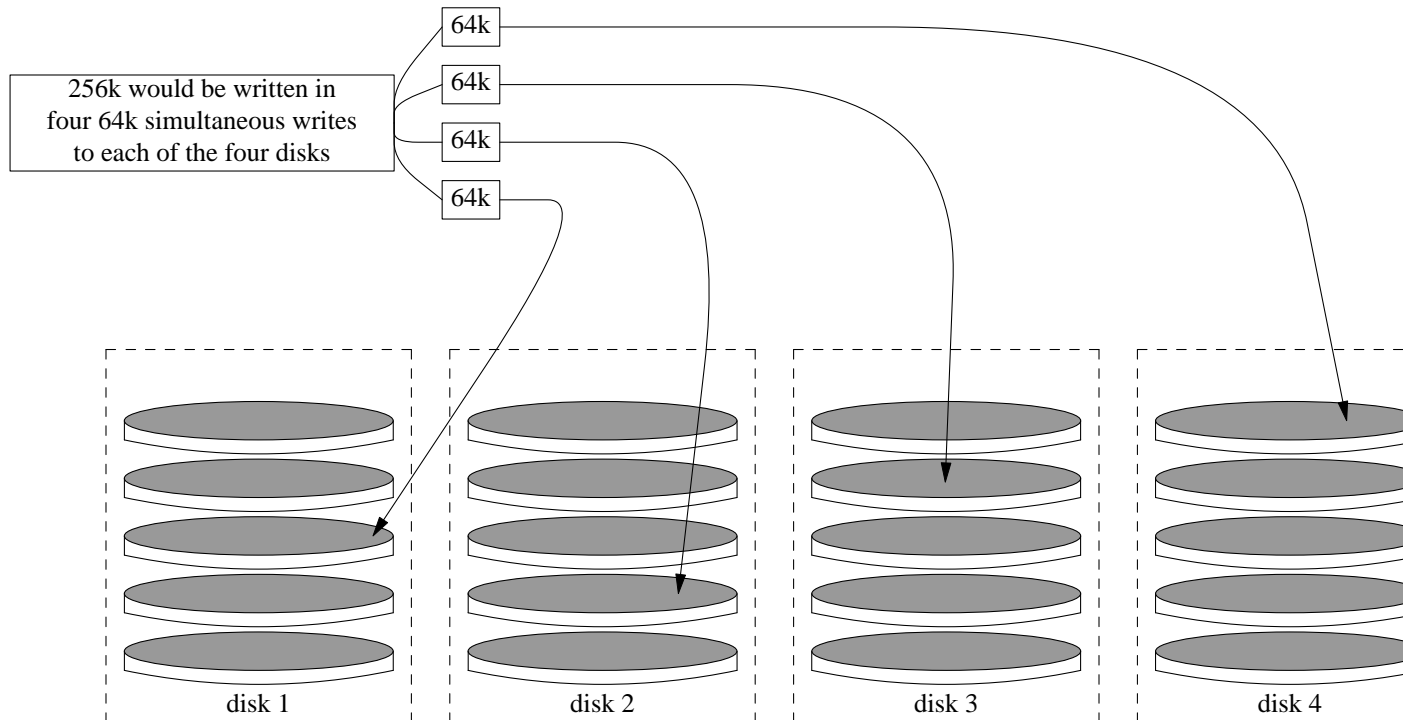
18.3. RAID0 - Striping

Written by Tom Rhodes i Murray Stokely.

Striping is a method used to combine several disk drives into a single volume. In many cases, this is done through the use of hardware controllers. The GEOM disk subsystem provides software support for RAID0, also known as disk striping.

In a RAID0 system, data are split up in blocks that get written across all the drives in the array. Instead of having to wait on the system to write 256k to one disk, a RAID0 system can simultaneously write 64k to each of four different disks, offering superior I/O performance. This performance can be enhanced further by using multiple disk controllers.

Each disk in a RAID0 stripe must be of the same size, since I/O requests are interleaved to read or write to multiple disks in parallel.



Creating a stripe of unformatted ATA disks

1. Load the `geom_stripe` module:

```
# kldload geom_stripe.ko
```
2. Ensure that a suitable mount point exists. If this volume will become a root partition, then temporarily use another mount point such as `/mnt`:

```
# mkdir /mnt
```
3. Determine the device names for the disks which will be striped, and create the new stripe device. For example, the following command could be used to stripe two unused, unpartitioned ATA disks: `/dev/ad2` and `/dev/ad3`.

```
# gstripe label -v st0 /dev/ad2 /dev/ad3
```
4. A partition table must be created on the new volume with the following command:

```
# bsdlabel -wB /dev/stripe/st0
```
5. This process should have created two other devices in the `/dev/stripe` directory in addition to the `st0` device. Those include `st0a` and `st0c`. A file system must now be created on the `st0a` device using the following `newfs` command:

```
# newfs -U /dev/stripe/st0a
```

Many numbers will glide across the screen, and after a few seconds, the process will be complete. The volume has been created and is ready to be mounted.

The following command can be used to manually mount a newly created disk stripe:

```
# mount /dev/stripe/st0a /mnt
```

To mount this striped file system automatically during the boot process, place the volume information in `/etc/fstab` file:

```
# echo "/dev/stripe/st0a /mnt ufs rw 2 2" \
  >> /etc/fstab
```

The `geom_stripe` module must also be automatically loaded during system initialization, by adding a line to `/boot/loader.conf`:

```
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

18.4. RAID1 - Mirroring

Mirroring is a technology used by many corporations and home users to back up data without interruption. When a mirror exists, it simply means that diskB replicates diskA. Or, perhaps diskC+D replicates diskA+B. Regardless of the disk configuration, the important aspect is that information on one disk or partition is being replicated. Later, that information could be more easily restored, backed up without causing service or access interruption, and even be physically stored in a data safe.

To begin, ensure the system has two disk drives of equal size, this exercise assumes they are direct access (da(4)) SCSI disks.

Begin by installing FreeBSD on the first disk with only two partitions. One should be a swap partition, double the RAM size and all remaining space devoted to the root (/) file system. It is possible to have separate partitions for other mount points; however, this will increase the difficulty level ten fold due to manual alteration of the `bsdlabeled(8)` and `fdisk(8)` settings.

Reboot and wait for the system to fully initialize. Once this process has completed, log in as the `root` user.

Create the `/dev/mirror/gm` device and link it with `/dev/da1`:

```
# gmirror label -vnb round-robin gm0 /dev/da1
```

The system should respond with:

```
Metadata value stored on /dev/da1.
Done.
```

Initialize GEOM, this will load the `/boot/kernel/geom_mirror.ko` kernel module:

```
# gmirror load
```

Notatka: This command should have created the `gm0`, device node under the `/dev/mirror` directory.

Install a generic `fdisk` label and boot code to newly created `gm0` device:

```
# fdisk -vBI /dev/mirror/gm0
```

Now install generic `bsdlabeled` information:

```
# bsdlabel -wB /dev/mirror/gm0s1
```

Notatka: If multiple slices and partitions exist, the flags for the previous two commands will require alteration. They must match the slice and partition size of the other disk.

Use the `newfs(8)` utility to create a default file system on the `gm0s1a` device node:

```
# newfs -U /dev/mirror/gm0s1a
```

This should have caused the system to spit out some information and a bunch of numbers. This is good. Examine the screen for any error messages and mount the device to the `/mnt` mount point:

```
# mount /dev/mirror/gm0s1a /mnt
```

Now move all data from the boot disk over to this new file system. This example uses the `dump(8)` and `restore(8)` commands; however, `dd(1)` would also work with this scenario.

```
# dump -L -0 -f- / | (cd /mnt && restore -r -v -f-)
```

This must be done for each file system. Simply place the appropriate file system in the correct location when running the aforementioned command.

Now edit the replicated `/mnt/etc/fstab` file and remove or comment out the swap file ¹. Change the other file system information to use the new disk. See the following example:

# Device	Mountpoint	FStype	Options	Dump	Pass#
#/dev/da0s2b	none	swap	sw	0	0
/dev/mirror/gm0s1a	/	ufs	rw	1	1

Now create a `boot.conf` file on both the current and new root partitions. This file will „help” the system BIOS boot the correct drive:

```
# echo "1:da(1,a)/boot/loader" > /boot.config
```

```
# echo "1:da(1,a)/boot/loader" > /mnt/boot.config
```

Notatka: We have placed it on both root partitions to ensure proper boot up. If for some reason the system cannot read from the new root partition, a failsafe is available.

Now add the following line to the new `/boot/loader.conf`:

```
# echo 'geom_mirror_load="YES"' >> /mnt/boot/loader.conf
```

This will instruct loader(8) utility to load the `geom_mirror.ko` module during system initialization.

Reboot the system:

```
# shutdown -r now
```

If all has gone well, the system should have booted from the `gm0s1a` device and a `login` prompt should be waiting. If something went wrong, see review the forthcoming troubleshooting section. Now add the `da0` disk to `gm0` device:

```
# gmirror configure -a gm0
# gmirror insert gm0 /dev/da0
```

The `-a` flag tells `gmirror(8)` to use automatic synchronization; i.e., mirror the disk writes automatically. The manual page explains how to rebuild and replace disks, although it uses `data` in place of `gm0`.

18.4.1. Troubleshooting

18.4.1.1. System refuses to boot

If the system boots up to a prompt similar to:

```
ffs_mountroot: can't find rootvp
Root mount failed: 6
mountroot>
```

Reboot the machine using the power or reset button. At the boot menu, select option six (6). This will drop the system to a loader(8) prompt. Load the kernel module manually:

```
OK? load geom_mirror.ko
OK? boot
```

If this works then for whatever reason the module was not being loaded properly. Place:

```
options GEOM_MIRROR
```

in the kernel configuration file, rebuild and reinstall. That should remedy this issue.

18.5. GEOM Gate Network Devices

GEOM supports the remote use of devices, such as disks, CD-ROMs, files, etc. through the use of the gate utilities. This is similar to NFS.

To begin, an exports file must be created. This file specifies who is permitted to access the exported resources and what level of access they are offered. For example, to export the forth slice on the first SCSI disk, the following `/etc/gg.exports` is more than adequate:

```
192.168.1.0/24 RW /dev/da0s4d
```

It will allow all hosts inside the private network access the file system on the `da0s4d` partition.

To export this device, ensure it is not currently mounted, and start the `ggated(8)` server daemon:

```
# ggated
```

Now to mount the device on the client machine, issue the following commands:

```
# ggatec create -o rw 192.168.1.1 /dev/da0s4d  
  
ggate0  
  
# mount /dev/ggate0 /mnt
```

From here on, the device may be accessed through the `/mnt` mount point.

Notatka: It should be pointed out that this will fail if the device is currently mounted on either the server machine or any other machine on the network.

When the device is no longer needed, it may be safely unmounted with the `umount(8)` command, similar to any other disk device.

Przypisy

1. It should be noted that commenting out the swap file entry in `fstab` will most likely require you to re-establish a different way of enabling swap space. Please refer to Sekcja 11.14 for more information.

Rozdział 19. The Vinum Volume Manager

Originally written by Greg Lehey.

19.1. Synopsis

No matter what disks you have, there are always potential problems:

- They can be too small.
- They can be too slow.
- They can be too unreliable.

One way some users safeguard themselves against such issues is through the use of multiple, and sometimes redundant, disks.

In addition to supporting various cards and controllers for hardware RAID systems, the base FreeBSD system includes the Vinum Volume Manager, a block device driver that implements virtual disk drives.

Vinum provides more flexibility, performance, and reliability than traditional disk storage, and implements RAID-0, RAID-1, and RAID-5 models both individually and in combination.

This chapter provides an overview of potential problems with traditional disk storage, and an introduction to the Vinum Volume Manager.

Notatka: Starting with FreeBSD 5, Vinum has been rewritten in order to fit into the GEOM architecture (Rozdział 18), retaining the original ideas, terminology, and on-disk metadata. This rewrite is called *gvinum* (for *GEOM vinum*). The following text usually refers to *Vinum* as an abstract name, regardless of the implementation variant. Any command invocations should now be done using the `gvinum` command, and the name of the kernel module has been changed from `vinum.ko` to `geom_vinum.ko`, and all device nodes reside under `/dev/gvinum` instead of `/dev/vinum`. As of FreeBSD 6, the old Vinum implementation is no longer available in the code base.

19.2. Disks Are Too Small

Vinum is a so-called *Volume Manager*, a virtual disk driver that addresses these three problems. Let us look at them in more detail. Various solutions to these problems have been proposed and implemented:

Disks are getting bigger, but so are data storage requirements. Often you will find you want a file system that is bigger than the disks you have available. Admittedly, this problem is not as acute as it was ten years ago, but it still exists. Some systems have solved this by creating an abstract device which stores its data on a number of disks.

19.3. Access Bottlenecks

Modern systems frequently need to access data in a highly concurrent manner. For example, large FTP or HTTP servers can maintain thousands of concurrent sessions and have multiple 100 Mbit/s connections to the outside world, well beyond the sustained transfer rate of most disks.

Current disk drives can transfer data sequentially at up to 70 MB/s, but this value is of little importance in an environment where many independent processes access a drive, where they may achieve only a fraction of these values. In such cases it is more interesting to view the problem from the viewpoint of the disk subsystem: the important parameter is the load that a transfer places on the subsystem, in other words the time for which a transfer occupies the drives involved in the transfer.

In any disk transfer, the drive must first position the heads, wait for the first sector to pass under the read head, and then perform the transfer. These actions can be considered to be atomic: it does not make any sense to interrupt them.

Consider a typical transfer of about 10 kB: the current generation of high-performance disks can position the heads in an average of 3.5 ms. The fastest drives spin at 15,000 rpm, so the average rotational latency (half a revolution) is 2 ms. At 70 MB/s, the transfer itself takes about 150 μ s, almost nothing compared to the positioning time. In such a case, the effective transfer rate drops to a little over 1 MB/s and is clearly highly dependent on the transfer size.

The traditional and obvious solution to this bottleneck is „more spindles”: rather than using one large disk, it uses several smaller disks with the same aggregate storage space. Each disk is capable of positioning and transferring independently, so the effective throughput increases by a factor close to the number of disks used.

The exact throughput improvement is, of course, smaller than the number of disks involved: although each drive is capable of transferring in parallel, there is no way to ensure that the requests are evenly distributed across the drives. Inevitably the load on one drive will be higher than on another.

The evenness of the load on the disks is strongly dependent on the way the data is shared across the drives. In the following discussion, it is convenient to think of the disk storage as a large number of data sectors which are addressable by number, rather like the pages in a book. The most obvious method is to divide the virtual disk into groups of consecutive sectors the size of the individual physical disks and store them in this manner, rather like taking a large book and tearing it into smaller sections. This method is called *concatenation* and has the advantage that the disks are not required to have any specific size relationships. It works well when the access to the virtual disk is spread evenly about its address space. When access is concentrated on a smaller area, the improvement is less marked. Rysunek 19-1 illustrates the sequence in which storage units are allocated in a concatenated organization.

Rysunek 19-1. Concatenated Organization

Disk 1	Disk 2	Disk 3	Disk 4
0	6	10	12
1	7	11	13
2	8		14
3	9		15
4			16
5			17

An alternative mapping is to divide the address space into smaller, equal-sized components and store them sequentially on different devices. For example, the first 256 sectors may be stored on the first disk, the next 256 sectors on the next disk and so on. After filling the last disk, the process repeats until the disks are full. This mapping is called *striping* or RAID-0¹. Striping requires somewhat more effort to locate the data, and it can cause additional I/O load where a transfer is spread over multiple disks, but it can also provide a more constant load across the disks. Rysunek 19-2 illustrates the sequence in which storage units are allocated in a striped organization.

Rysunek 19-2. Striped Organization

Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23

19.4. Data Integrity

The final problem with current disks is that they are unreliable. Although disk drive reliability has increased tremendously over the last few years, they are still the most likely core component of a server to fail. When they do, the results can be catastrophic: replacing a failed disk drive and restoring data to it can take days.

The traditional way to approach this problem has been *mirroring*, keeping two copies of the data on different physical hardware. Since the advent of the RAID levels, this technique has also been called RAID level 1 or RAID-1. Any write to the volume writes to both locations; a read can be satisfied from either, so if one drive fails, the data is still available on the other drive.

Mirroring has two problems:

- The price. It requires twice as much disk storage as a non-redundant solution.
- The performance impact. Writes must be performed to both drives, so they take up twice the bandwidth of a non-mirrored volume. Reads do not suffer from a performance penalty: it even looks as if they are faster.

An alternative solution is *parity*, implemented in the RAID levels 2, 3, 4 and 5. Of these, RAID-5 is the most interesting. As implemented in Vinum, it is a variant on a striped organization which dedicates one block of each stripe to parity of the other blocks. As implemented by Vinum, a RAID-5 plex is similar to a striped plex, except that

it implements RAID-5 by including a parity block in each stripe. As required by RAID-5, the location of this parity block changes from one stripe to the next. The numbers in the data blocks indicate the relative block numbers.

Rysunek 19-3. RAID-5 Organization

Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	Parity
3	4	Parity	5
6	Parity	7	8
Parity	9	10	11
12	13	14	Parity
15	16	Parity	17

Compared to mirroring, RAID-5 has the advantage of requiring significantly less storage space. Read access is similar to that of striped organizations, but write access is significantly slower, approximately 25% of the read performance. If one drive fails, the array can continue to operate in degraded mode: a read from one of the remaining accessible drives continues normally, but a read from the failed drive is recalculated from the corresponding block from all the remaining drives.

19.5. Vinum Objects

In order to address these problems, Vinum implements a four-level hierarchy of objects:

- The most visible object is the virtual disk, called a *volume*. Volumes have essentially the same properties as a UNIX disk drive, though there are some minor differences. They have no size limitations.
- Volumes are composed of *plexes*, each of which represent the total address space of a volume. This level in the hierarchy thus provides redundancy. Think of plexes as individual disks in a mirrored array, each containing the same data.
- Since Vinum exists within the UNIX disk storage framework, it would be possible to use UNIX partitions as the building block for multi-disk plexes, but in fact this turns out to be too inflexible: UNIX disks can have only a limited number of partitions. Instead, Vinum subdivides a single UNIX partition (the *drive*) into contiguous areas called *subdisks*, which it uses as building blocks for plexes.
- Subdisks reside on Vinum *drives*, currently UNIX partitions. Vinum drives can contain any number of subdisks. With the exception of a small area at the beginning of the drive, which is used for storing configuration and state information, the entire drive is available for data storage.

The following sections describe the way these objects provide the functionality required of Vinum.

19.5.1. Volume Size Considerations

Plexes can include multiple subdisks spread over all drives in the Vinum configuration. As a result, the size of an individual drive does not limit the size of a plex, and thus of a volume.

19.5.2. Redundant Data Storage

Vinum implements mirroring by attaching multiple plexes to a volume. Each plex is a representation of the data in a volume. A volume may contain between one and eight plexes.

Although a plex represents the complete data of a volume, it is possible for parts of the representation to be physically missing, either by design (by not defining a subdisk for parts of the plex) or by accident (as a result of the failure of a drive). As long as at least one plex can provide the data for the complete address range of the volume, the volume is fully functional.

19.5.3. Performance Issues

Vinum implements both concatenation and striping at the plex level:

- A *concatenated plex* uses the address space of each subdisk in turn.
- A *striped plex* stripes the data across each subdisk. The subdisks must all have the same size, and there must be at least two subdisks in order to distinguish it from a concatenated plex.

19.5.4. Which Plex Organization?

The version of Vinum supplied with FreeBSD 9.1 implements two kinds of plex:

- Concatenated plexes are the most flexible: they can contain any number of subdisks, and the subdisks may be of different length. The plex may be extended by adding additional subdisks. They require less CPU time than striped plexes, though the difference in CPU overhead is not measurable. On the other hand, they are most susceptible to hot spots, where one disk is very active and others are idle.
- The greatest advantage of striped (RAID-0) plexes is that they reduce hot spots: by choosing an optimum sized stripe (about 256 kB), you can even out the load on the component drives. The disadvantages of this approach are (fractionally) more complex code and restrictions on subdisks: they must be all the same size, and extending a plex by adding new subdisks is so complicated that Vinum currently does not implement it. Vinum imposes an additional, trivial restriction: a striped plex must have at least two subdisks, since otherwise it is indistinguishable from a concatenated plex.

Tabela 19-1 summarizes the advantages and disadvantages of each plex organization.

Tabela 19-1. Vinum Plex Organizations

Plex type	Minimum subdisks	Can add subdisks	Must be equal size	Application
concatenated	1	yes	no	Large data storage with maximum placement flexibility and moderate performance

Plex type	Minimum subdisks	Can add subdisks	Must be equal size	Application
striped	2	no	yes	High performance in combination with highly concurrent access

19.6. Some Examples

Vinum maintains a *configuration database* which describes the objects known to an individual system. Initially, the user creates the configuration database from one or more configuration files with the aid of the `gvinum(8)` utility program. Vinum stores a copy of its configuration database on each disk slice (which Vinum calls a *device*) under its control. This database is updated on each state change, so that a restart accurately restores the state of each Vinum object.

19.6.1. The Configuration File

The configuration file describes individual Vinum objects. The definition of a simple volume might be:

```
drive a device /dev/da3h
volume myvol
  plex org concat
    sd length 512m drive a
```

This file describes four Vinum objects:

- The *drive* line describes a disk partition (*drive*) and its location relative to the underlying hardware. It is given the symbolic name *a*. This separation of the symbolic names from the device names allows disks to be moved from one location to another without confusion.
- The *volume* line describes a volume. The only required attribute is the name, in this case *myvol*.
- The *plex* line defines a plex. The only required parameter is the organization, in this case *concat*. No name is necessary: the system automatically generates a name from the volume name by adding the suffix *.px*, where *x* is the number of the plex in the volume. Thus this plex will be called *myvol.p0*.
- The *sd* line describes a subdisk. The minimum specifications are the name of a drive on which to store it, and the length of the subdisk. As with plexes, no name is necessary: the system automatically assigns names derived from the plex name by adding the suffix *.sx*, where *x* is the number of the subdisk in the plex. Thus Vinum gives this subdisk the name *myvol.p0.s0*.

After processing this file, `gvinum(8)` produces the following output:

```
# gvinum -> create config1
Configuration summary
Drives:      1 (4 configured)
Volumes:     1 (4 configured)
Plexes:      1 (8 configured)
Subdisks:    1 (16 configured)

D a          State: up      Device /dev/da3h      Avail: 2061/2573 MB (80%)

V myvol      State: up      Plexes:      1 Size:    512 MB
```

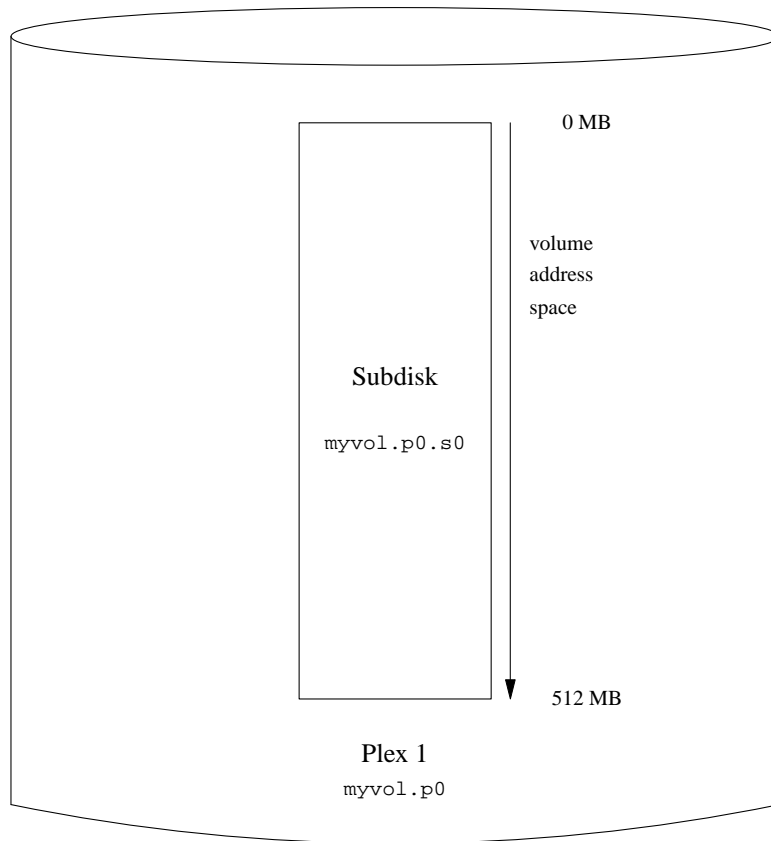
```

P myvol.p0          C State: up      Subdisks:    1 Size:      512 MB
S myvol.p0.s0       State: up        PO:          0 B Size:    512 MB

```

This output shows the brief listing format of `gvinum(8)`. It is represented graphically in Rysunek 19-4.

Rysunek 19-4. A Simple Vinum Volume



This figure, and the ones which follow, represent a volume, which contains the plexes, which in turn contain the subdisks. In this trivial example, the volume contains one plex, and the plex contains one subdisk.

This particular volume has no specific advantage over a conventional disk partition. It contains a single plex, so it is not redundant. The plex contains a single subdisk, so there is no difference in storage allocation from a conventional disk partition. The following sections illustrate various more interesting configuration methods.

19.6.2. Increased Resilience: Mirroring

The resilience of a volume can be increased by mirroring. When laying out a mirrored volume, it is important to ensure that the subdisks of each plex are on different drives, so that a drive failure will not take down both plexes. The following configuration mirrors a volume:

```

drive b device /dev/da4h
volume mirror
plex org concat
sd length 512m drive a
plex org concat
sd length 512m drive b

```

In this example, it was not necessary to specify a definition of drive *a* again, since Vinum keeps track of all objects in its configuration database. After processing this definition, the configuration looks like:

```

Drives:      2 (4 configured)
Volumes:     2 (4 configured)
Plexes:      3 (8 configured)
Subdisks:    3 (16 configured)

D a          State: up      Device /dev/da3h    Avail: 1549/2573 MB (60%)
D b          State: up      Device /dev/da4h    Avail: 2061/2573 MB (80%)

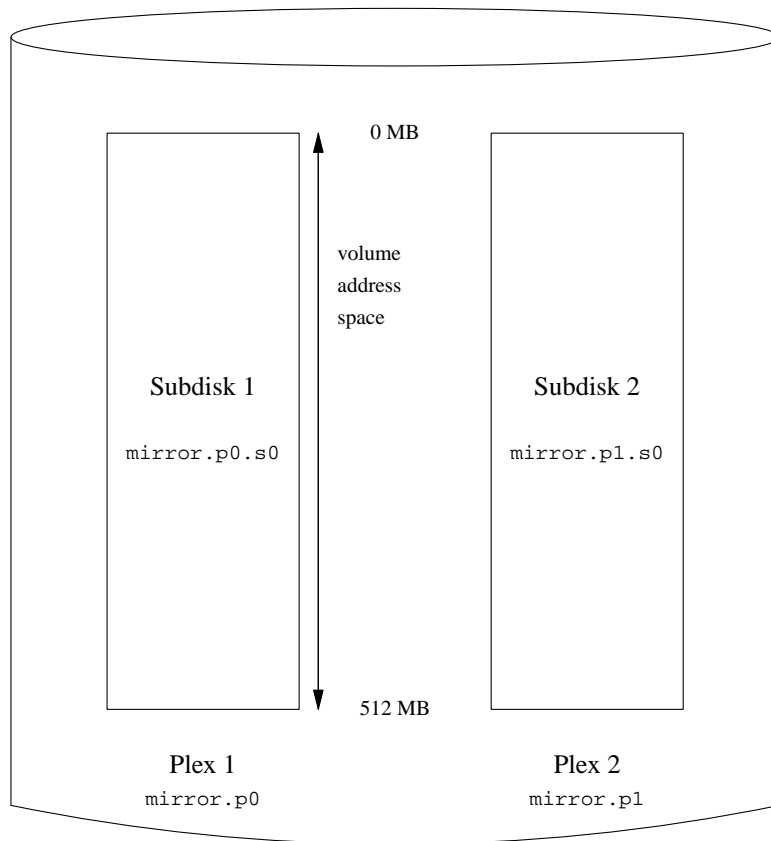
V myvol      State: up      Plexes:      1 Size:    512 MB
V mirror     State: up      Plexes:      2 Size:    512 MB

P myvol.p0   C State: up      Subdisks:    1 Size:    512 MB
P mirror.p0  C State: up      Subdisks:    1 Size:    512 MB
P mirror.pl  C State: initializing Subdisks:    1 Size:    512 MB

S myvol.p0.s0 State: up      PO:      0 B Size:    512 MB
S mirror.p0.s0 State: up      PO:      0 B Size:    512 MB
S mirror.pl.s0 State: empty   PO:      0 B Size:    512 MB

```

Rysunek 19-5 shows the structure graphically.

Rysunek 19-5. A Mirrored Vinum Volume

In this example, each plex contains the full 512 MB of address space. As in the previous example, each plex contains only a single subdisk.

19.6.3. Optimizing Performance

The mirrored volume in the previous example is more resistant to failure than an unmirrored volume, but its performance is less: each write to the volume requires a write to both drives, using up a greater proportion of the total disk bandwidth. Performance considerations demand a different approach: instead of mirroring, the data is striped across as many disk drives as possible. The following configuration shows a volume with a plex striped across four disk drives:

```
drive c device /dev/da5h
drive d device /dev/da6h
volume stripe
plex org striped 512k
sd length 128m drive a
```

```
sd length 128m drive b
sd length 128m drive c
sd length 128m drive d
```

As before, it is not necessary to define the drives which are already known to Vinum. After processing this definition, the configuration looks like:

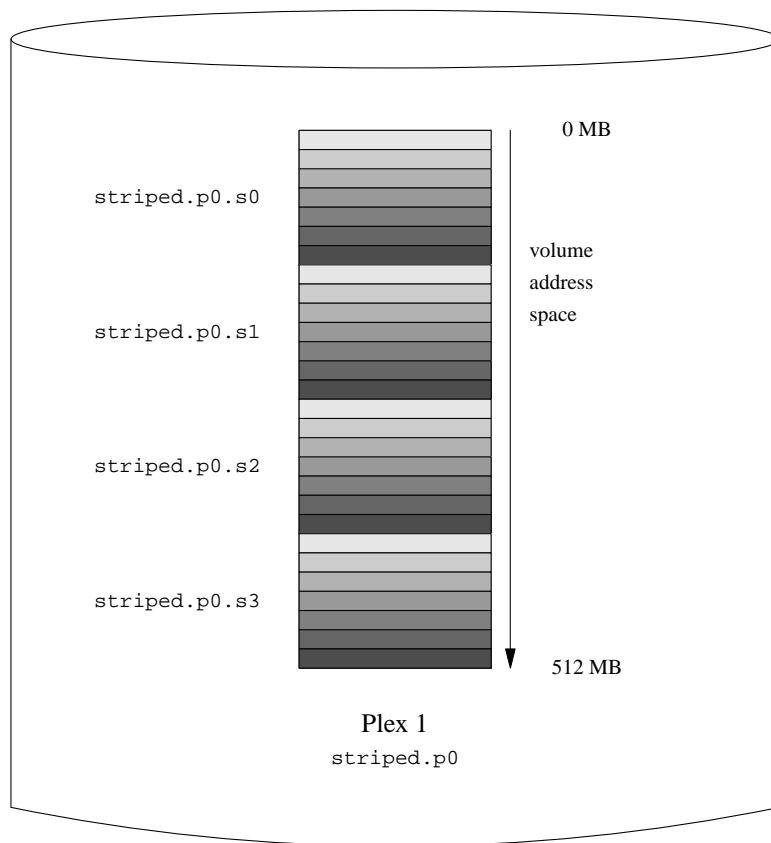
```
Drives:      4 (4 configured)
Volumes:     3 (4 configured)
Plexes:      4 (8 configured)
Subdisks:    7 (16 configured)

D a          State: up      Device /dev/da3h      Avail: 1421/2573 MB (55%)
D b          State: up      Device /dev/da4h      Avail: 1933/2573 MB (75%)
D c          State: up      Device /dev/da5h      Avail: 2445/2573 MB (95%)
D d          State: up      Device /dev/da6h      Avail: 2445/2573 MB (95%)

V myvol      State: up      Plexes:      1 Size:      512 MB
V mirror     State: up      Plexes:      2 Size:      512 MB
V striped    State: up      Plexes:      1 Size:      512 MB

P myvol.p0   C State: up      Subdisks:    1 Size:      512 MB
P mirror.p0  C State: up      Subdisks:    1 Size:      512 MB
P mirror.p1  C State: initializing Subdisks:    1 Size:      512 MB
P striped.p1  State: up      Subdisks:    1 Size:      512 MB

S myvol.p0.s0 State: up      PO:         0 B Size:      512 MB
S mirror.p0.s0 State: up      PO:         0 B Size:      512 MB
S mirror.p1.s0 State: empty   PO:         0 B Size:      512 MB
S striped.p0.s0 State: up      PO:         0 B Size:      128 MB
S striped.p0.s1 State: up      PO:         512 kB Size:      128 MB
S striped.p0.s2 State: up      PO:        1024 kB Size:      128 MB
S striped.p0.s3 State: up      PO:        1536 kB Size:      128 MB
```

Rysunek 19-6. A Striped Vinum Volume

This volume is represented in Rysunek 19-6. The darkness of the stripes indicates the position within the plex address space: the lightest stripes come first, the darkest last.

19.6.4. Resilience and Performance

With sufficient hardware, it is possible to build volumes which show both increased resilience and increased performance compared to standard UNIX partitions. A typical configuration file might be:

```

volume raid10
plex org striped 512k
  sd length 102480k drive a
  sd length 102480k drive b
  sd length 102480k drive c
  sd length 102480k drive d
  sd length 102480k drive e
plex org striped 512k
  sd length 102480k drive c

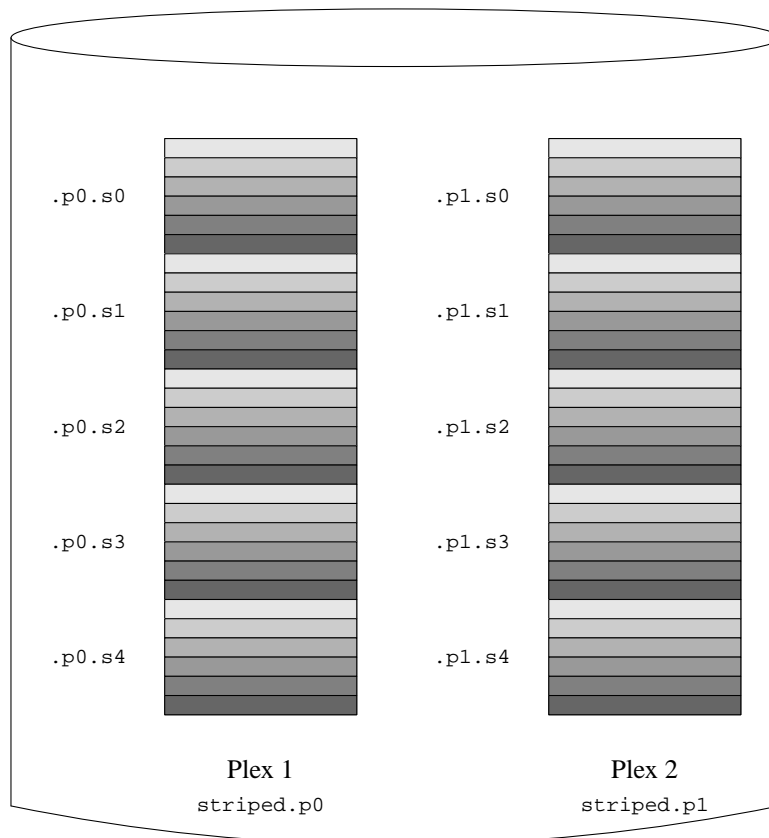
```

```
sd length 102480k drive d
sd length 102480k drive e
sd length 102480k drive a
sd length 102480k drive b
```

The subdisks of the second plex are offset by two drives from those of the first plex: this helps ensure that writes do not go to the same subdisks even if a transfer goes over two drives.

Rysunek 19-7 represents the structure of this volume.

Rysunek 19-7. A Mirrored, Striped Vinum Volume



19.7. Object Naming

As described above, Vinum assigns default names to plexes and subdisks, although they may be overridden. Overriding the default names is not recommended: experience with the VERITAS volume manager, which allows

arbitrary naming of objects, has shown that this flexibility does not bring a significant advantage, and it can cause confusion.

Names may contain any non-blank character, but it is recommended to restrict them to letters, digits and the underscore characters. The names of volumes, plexes and subdisks may be up to 64 characters long, and the names of drives may be up to 32 characters long.

Vinum objects are assigned device nodes in the hierarchy `/dev/gvinum`. The configuration shown above would cause Vinum to create the following device nodes:

-

Notatka: This only applies to the historic Vinum implementation.

The control devices `/dev/vinum/control` and `/dev/vinum/controld`, which are used by `gvinum(8)` and the Vinum daemon respectively.

- Device entries for each volume. These are the main devices used by Vinum. Thus the configuration above would include the devices `/dev/gvinum/myvol`, `/dev/gvinum/mirror`, `/dev/gvinum/striped`, `/dev/gvinum/raid5` and `/dev/gvinum/raid10`.

-

Notatka: This only applies to the historic Vinum implementation.

A directory `/dev/vinum/drive` with entries for each drive. These entries are in fact symbolic links to the corresponding disk nodes.

- All volumes get direct entries under `/dev/gvinum/`.
- The directories `/dev/gvinum/plex`, and `/dev/gvinum/sd`, which contain device nodes for each plex and for each subdisk, respectively.

For example, consider the following configuration file:

```
drive drive1 device /dev/sd1h
drive drive2 device /dev/sd2h
drive drive3 device /dev/sd3h
drive drive4 device /dev/sd4h
volume s64 setupstate
plex org striped 64k
sd length 100m drive drive1
sd length 100m drive drive2
sd length 100m drive drive3
sd length 100m drive drive4
```

After processing this file, `gvinum(8)` creates the following structure in `/dev/gvinum`:

```
drwxr-xr-x  2 root  wheel      512 Apr 13 16:46 plex
```

```

crwxr-xr--  1 root  wheel   91,   2 Apr 13 16:46 s64
drwxr-xr-x  2 root  wheel   512 Apr 13 16:46 sd

/dev/vinum/plex:
total 0
crwxr-xr--  1 root  wheel   25, 0x10000002 Apr 13 16:46 s64.p0

/dev/vinum/sd:
total 0
crwxr-xr--  1 root  wheel   91, 0x20000002 Apr 13 16:46 s64.p0.s0
crwxr-xr--  1 root  wheel   91, 0x20100002 Apr 13 16:46 s64.p0.s1
crwxr-xr--  1 root  wheel   91, 0x20200002 Apr 13 16:46 s64.p0.s2
crwxr-xr--  1 root  wheel   91, 0x20300002 Apr 13 16:46 s64.p0.s3

```

Although it is recommended that plexes and subdisks should not be allocated specific names, Vinum drives must be named. This makes it possible to move a drive to a different location and still recognize it automatically. Drive names may be up to 32 characters long.

19.7.1. Creating File Systems

Volumes appear to the system to be identical to disks, with one exception. Unlike UNIX drives, Vinum does not partition volumes, which thus do not contain a partition table. This has required modification to some disk utilities, notably `newfs(8)`, which previously tried to interpret the last letter of a Vinum volume name as a partition identifier. For example, a disk drive may have a name like `/dev/ad0a` or `/dev/da2h`. These names represent the first partition (a) on the first (0) IDE disk (ad) and the eighth partition (h) on the third (2) SCSI disk (da) respectively. By contrast, a Vinum volume might be called `/dev/gvinum/concat`, a name which has no relationship with a partition name.

Normally, `newfs(8)` interprets the name of the disk and complains if it cannot understand it. For example:

```

# newfs /dev/gvinum/concat
newfs: /dev/gvinum/concat: can't figure out file system partition

```

In order to create a file system on this volume, use `newfs(8)`:

```

# newfs /dev/gvinum/concat

```

Notatka: On FreeBSD versions prior to 5.0 `newfs(8)` requires an additional `-v` flag and the old device naming scheme:

```

# newfs -v /dev/vinum/concat

```

19.8. Configuring Vinum

The `GENERIC` kernel does not contain Vinum. It is possible to build a special kernel which includes Vinum, but this is not recommended. The standard way to start Vinum is as a kernel module (`kld`). You do not even need to use `kldload(8)` for Vinum: when you start `gvinum(8)`, it checks whether the module has been loaded, and if it is not, it loads it automatically.

19.8.1. Startup

Vinum stores configuration information on the disk slices in essentially the same form as in the configuration files. When reading from the configuration database, Vinum recognizes a number of keywords which are not allowed in the configuration files. For example, a disk configuration might contain the following text:

```
volume myvol state up
volume bigraid state down
plex name myvol.p0 state up org concat vol myvol
plex name myvol.p1 state up org concat vol myvol
plex name myvol.p2 state init org striped 512b vol myvol
plex name bigraid.p0 state initializing org raid5 512b vol bigraid
sd name myvol.p0.s0 drive a plex myvol.p0 state up len 1048576b driveoffset 265b plexoffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b driveoffset 265b plexoffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b driveoffset 265b plexoffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b driveoffset 265b plexoffset 1048576b
sd name myvol.p2.s0 drive a plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 0b
sd name myvol.p2.s1 drive b plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 524288b
sd name myvol.p2.s2 drive c plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 1048576b
sd name myvol.p2.s3 drive d plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 1572864b
sd name bigraid.p0.s0 drive a plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 0b
sd name bigraid.p0.s1 drive b plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 4194304b
sd name bigraid.p0.s2 drive c plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 8388608b
sd name bigraid.p0.s3 drive d plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 12582912b
sd name bigraid.p0.s4 drive e plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 16777216b
```

The obvious differences here are the presence of explicit location information and naming (both of which are also allowed, but discouraged, for use by the user) and the information on the states (which are not available to the user). Vinum does not store information about drives in the configuration information: it finds the drives by scanning the configured disk drives for partitions with a Vinum label. This enables Vinum to identify drives correctly even if they have been assigned different UNIX drive IDs.

19.8.1.1. Automatic Startup

Notatka: This information only relates to the historic Vinum implementation. *Gvinum* always features an automatic startup once the kernel module is loaded.

In order to start Vinum automatically when you boot the system, ensure that you have the following line in your `/etc/rc.conf`:

```
start_vinum="YES"                # set to YES to start vinum
```

If you do not have a file `/etc/rc.conf`, create one with this content. This will cause the system to load the Vinum kld at startup, and to start any objects mentioned in the configuration. This is done before mounting file systems, so it is possible to automatically `fsck(8)` and mount file systems on Vinum volumes.

When you start Vinum with the `vinum start` command, Vinum reads the configuration database from one of the Vinum drives. Under normal circumstances, each drive contains an identical copy of the configuration database, so it does not matter which drive is read. After a crash, however, Vinum must determine which drive was updated most recently and read the configuration from this drive. It then updates the configuration if necessary from progressively older drives.

19.9. Using Vinum for the Root Filesystem

For a machine that has fully-mirrored filesystems using Vinum, it is desirable to also mirror the root filesystem. Setting up such a configuration is less trivial than mirroring an arbitrary filesystem because:

- The root filesystem must be available very early during the boot process, so the Vinum infrastructure must already be available at this time.
- The volume containing the root filesystem also contains the system bootstrap and the kernel, which must be read using the host system's native utilities (e. g. the BIOS on PC-class machines) which often cannot be taught about the details of Vinum.

In the following sections, the term „root volume” is generally used to describe the Vinum volume that contains the root filesystem. It is probably a good idea to use the name "root" for this volume, but this is not technically required in any way. All command examples in the following sections assume this name though.

19.9.1. Starting up Vinum Early Enough for the Root Filesystem

There are several measures to take for this to happen:

- Vinum must be available in the kernel at boot-time. Thus, the method to start Vinum automatically described in Sekcja 19.8.1.1 is not applicable to accomplish this task, and the `start_vinum` parameter must actually *not* be set when the following setup is being arranged. The first option would be to compile Vinum statically into the kernel, so it is available all the time, but this is usually not desirable. There is another option as well, to have `/boot/loader` (Sekcja 12.3.3) load the vinum kernel module early, before starting the kernel. This can be accomplished by putting the line:

```
geom_vinum_load="YES"
```

into the file `/boot/loader.conf`.

•

Notatka: For *Gvinum*, all startup is done automatically once the kernel module has been loaded, so the procedure described above is all that is needed. The following text documents the behaviour of the historic Vinum system, for the sake of older setups.

Vinum must be initialized early since it needs to supply the volume for the root filesystem. By default, the Vinum kernel part is not looking for drives that might contain Vinum volume information until the administrator (or one of the startup scripts) issues a `vinum start` command.

Notatka: The following paragraphs are outlining the steps needed for FreeBSD 5.X and above. The setup required for FreeBSD 4.X differs, and is described below in Sekcja 19.9.5.

By placing the line:

```
vinum.autostart="YES"
```

into `/boot/loader.conf`, Vinum is instructed to automatically scan all drives for Vinum information as part of the kernel startup.

Note that it is not necessary to instruct the kernel where to look for the root filesystem. `/boot/loader` looks up the name of the root device in `/etc/fstab`, and passes this information on to the kernel. When it comes to mount the root filesystem, the kernel figures out from the device name provided which driver to ask to translate this into the internal device ID (major/minor number).

19.9.2. Making a Vinum-based Root Volume Accessible to the Bootstrap

Since the current FreeBSD bootstrap is only 7.5 KB of code, and already has the burden of reading files (like `/boot/loader`) from the UFS filesystem, it is sheer impossible to also teach it about internal Vinum structures so it could parse the Vinum configuration data, and figure out about the elements of a boot volume itself. Thus, some tricks are necessary to provide the bootstrap code with the illusion of a standard "a" partition that contains the root filesystem.

For this to be possible at all, the following requirements must be met for the root volume:

- The root volume must not be striped or RAID-5.
- The root volume must not contain more than one concatenated subdisk per plex.

Note that it is desirable and possible that there are multiple plexes, each containing one replica of the root filesystem. The bootstrap process will, however, only use one of these replica for finding the bootstrap and all the files, until the kernel will eventually mount the root filesystem itself. Each single subdisk within these plexes will then need its own "a" partition illusion, for the respective device to become bootable. It is not strictly needed that each of these faked "a" partitions is located at the same offset within its device, compared with other devices containing plexes of the root volume. However, it is probably a good idea to create the Vinum volumes that way so the resulting mirrored devices are symmetric, to avoid confusion.

In order to set up these "a" partitions, for each device containing part of the root volume, the following needs to be done:

1. The location (offset from the beginning of the device) and size of this device's subdisk that is part of the root volume need to be examined, using the command:

```
# gvinum l -rv root
```

Note that Vinum offsets and sizes are measured in bytes. They must be divided by 512 in order to obtain the block numbers that are to be used in the `bsdlabel` command.

2. Run the command:

```
# bsdlabel -e devname
```

for each device that participates in the root volume. *devname* must be either the name of the disk (like `da0`) for disks without a slice (aka. `fdisk`) table, or the name of the slice (like `ad0s1`).

If there is already an "a" partition on the device (presumably, containing a pre-Vinum root filesystem), it should be renamed to something else, so it remains accessible (just in case), but will no longer be used by default to bootstrap the system. Note that active partitions (like a root filesystem currently mounted) cannot be renamed, so this must be executed either when being booted from a „Fixit” medium, or in a two-step process, where (in a mirrored situation) the disk that has not been currently booted is being manipulated first.

Then, the offset the Vinum partition on this device (if any) must be added to the offset of the respective root volume subdisk on this device. The resulting value will become the "offset" value for the new "a" partition. The "size" value for this partition can be taken verbatim from the calculation above. The "fstype" should be 4.2BSD. The "fsize", "bsize", and "cpb" values should best be chosen to match the actual filesystem, though they are fairly unimportant within this context.

That way, a new "a" partition will be established that overlaps the Vinum partition on this device. Note that the `bsdlabeled` will only allow for this overlap if the Vinum partition has properly been marked using the "vinum" fstype.

3. That's all! A faked "a" partition does exist now on each device that has one replica of the root volume. It is highly recommendable to verify the result again, using a command like:

```
# fsck -n /dev/devnamea
```

It should be remembered that all files containing control information must be relative to the root filesystem in the Vinum volume which, when setting up a new Vinum root volume, might not match the root filesystem that is currently active. So in particular, the files `/etc/fstab` and `/boot/loader.conf` need to be taken care of.

At next reboot, the bootstrap should figure out the appropriate control information from the new Vinum-based root filesystem, and act accordingly. At the end of the kernel initialization process, after all devices have been announced, the prominent notice that shows the success of this setup is a message like:

```
Mounting root from ufs:/dev/gvinum/root
```

19.9.3. Example of a Vinum-based Root Setup

After the Vinum root volume has been set up, the output of `gvinum l -rv root` could look like:

```
...
Subdisk root.p0.s0:
    Size:          125829120 bytes (120 MB)
    State: up
    Plex root.p0 at offset 0 (0 B)
    Drive disk0 (/dev/da0h) at offset 135680 (132 kB)

Subdisk root.p1.s0:
    Size:          125829120 bytes (120 MB)
    State: up
    Plex root.p1 at offset 0 (0 B)
    Drive disk1 (/dev/dal1h) at offset 135680 (132 kB)
```

The values to note are 135680 for the offset (relative to partition `/dev/da0h`). This translates to 265 512-byte disk blocks in `bsdlabeled`'s terms. Likewise, the size of this root volume is 245760 512-byte blocks. `/dev/dal1h`, containing the second replica of this root volume, has a symmetric setup.

The `bsdlabeled` for these devices might look like:

```
...
8 partitions:
#      size  offset  fstype  [fsize bsize bps/cpb]
a:    245760    281    4.2BSD    2048 16384    0  # (Cyl.    0*- 15*)
```

```

c: 71771688      0      unused      0      0      # (Cyl.      0 - 4467*)
h: 71771672      16      vinum        # (Cyl.      0*- 4467*)

```

It can be observed that the "size" parameter for the faked "a" partition matches the value outlined above, while the "offset" parameter is the sum of the offset within the Vinum partition "h", and the offset of this partition within the device (or slice). This is a typical setup that is necessary to avoid the problem described in Sekcja 19.9.4.3. It can also be seen that the entire "a" partition is completely within the "h" partition containing all the Vinum data for this device.

Note that in the above example, the entire device is dedicated to Vinum, and there is no leftover pre-Vinum root partition, since this has been a newly set-up disk that was only meant to be part of a Vinum configuration, ever.

19.9.4. Troubleshooting

If something goes wrong, a way is needed to recover from the situation. The following list contains few known pitfalls and solutions.

19.9.4.1. System Bootstrap Loads, but System Does Not Boot

If for any reason the system does not continue to boot, the bootstrap can be interrupted with by pressing the **space** key at the 10-seconds warning. The loader variables (like `vinum.autostart`) can be examined using the `show`, and manipulated using `set` or `unset` commands.

If the only problem was that the Vinum kernel module was not yet in the list of modules to load automatically, a simple `load geom_vinum` will help.

When ready, the boot process can be continued with a `boot -as`. The options `-as` will request the kernel to ask for the root filesystem to mount (`-a`), and make the boot process stop in single-user mode (`-s`), where the root filesystem is mounted read-only. That way, even if only one plex of a multi-plex volume has been mounted, no data inconsistency between plexes is being risked.

At the prompt asking for a root filesystem to mount, any device that contains a valid root filesystem can be entered. If `/etc/fstab` had been set up correctly, the default should be something like `ufs:/dev/gvinum/root`. A typical alternate choice would be something like `ufs:da0d` which could be a hypothetical partition that contains the pre-Vinum root filesystem. Care should be taken if one of the alias "a" partitions are entered here that are actually reference to the subdisks of the Vinum root device, because in a mirrored setup, this would only mount one piece of a mirrored root device. If this filesystem is to be mounted read-write later on, it is necessary to remove the other plex(es) of the Vinum root volume since these plexes would otherwise carry inconsistent data.

19.9.4.2. Only Primary Bootstrap Loads

If `/boot/loader` fails to load, but the primary bootstrap still loads (visible by a single dash in the left column of the screen right after the boot process starts), an attempt can be made to interrupt the primary bootstrap at this point, using the **space** key. This will make the bootstrap stop in stage two, see Sekcja 12.3.2. An attempt can be made here to boot off an alternate partition, like the partition containing the previous root filesystem that has been moved away from "a" above.

19.9.4.3. Nothing Boots, the Bootstrap Panics

This situation will happen if the bootstrap had been destroyed by the Vinum installation. Unfortunately, Vinum accidentally currently leaves only 4 KB at the beginning of its partition free before starting to write its Vinum header information. However, the stage one and two bootstraps plus the bsdlable embedded between them currently require 8 KB. So if a Vinum partition was started at offset 0 within a slice or disk that was meant to be bootable, the Vinum setup will trash the bootstrap.

Similarly, if the above situation has been recovered, for example by booting from a „Fixit” medium, and the bootstrap has been re-installed using `bsdlablel -B` as described in Sekcja 12.3.2, the bootstrap will trash the Vinum header, and Vinum will no longer find its disk(s). Though no actual Vinum configuration data or data in Vinum volumes will be trashed by this, and it would be possible to recover all the data by entering exact the same Vinum configuration data again, the situation is hard to fix at all. It would be necessary to move the entire Vinum partition by at least 4 KB off, in order to have the Vinum header and the system bootstrap no longer collide.

19.9.5. Differences for FreeBSD 4.X

Under FreeBSD 4.X, some internal functions required to make Vinum automatically scan all disks are missing, and the code that figures out the internal ID of the root device is not smart enough to handle a name like `/dev/vinum/root` automatically. Therefore, things are a little different here.

Vinum must explicitly be told which disks to scan, using a line like the following one in `/boot/loader.conf`:

```
vinum.drives="/dev/da0 /dev/da1"
```

It is important that all drives are mentioned that could possibly contain Vinum data. It does not harm if *more* drives are listed, nor is it necessary to add each slice and/or partition explicitly, since Vinum will scan all slices and partitions of the named drives for valid Vinum headers.

Since the routines used to parse the name of the root filesystem, and derive the device ID (major/minor number) are only prepared to handle „classical” device names like `/dev/ad0s1a`, they cannot make any sense out of a root volume name like `/dev/vinum/root`. For that reason, Vinum itself needs to pre-setup the internal kernel parameter that holds the ID of the root device during its own initialization. This is requested by passing the name of the root volume in the loader variable `vinum.root`. The entry in `/boot/loader.conf` to accomplish this looks like:

```
vinum.root="root"
```

Now, when the kernel initialization tries to find out the root device to mount, it sees whether some kernel module has already pre-initialized the kernel parameter for it. If that is the case, *and* the device claiming the root device matches the major number of the driver as figured out from the name of the root device string being passed (that is, „vinum” in our case), it will use the pre-allocated device ID, instead of trying to figure out one itself. That way, during the usual automatic startup, it can continue to mount the Vinum root volume for the root filesystem.

However, when `boot -a` has been requesting to ask for entering the name of the root device manually, it must be noted that this routine still cannot actually parse a name entered there that refers to a Vinum volume. If any device name is entered that does not refer to a Vinum device, the mismatch between the major numbers of the pre-allocated root parameter and the driver as figured out from the given name will make this routine enter its normal parser, so entering a string like `ufs:da0d` will work as expected. Note that if this fails, it is however no longer possible to re-enter a string like `ufs:vinum/root` again, since it cannot be parsed. The only way out is to reboot again, and start over then. (At the „askroot” prompt, the initial `/dev/` can always be omitted.)

Przypisy

1. RAID stands for *Redundant Array of Inexpensive Disks* and offers various forms of fault tolerance, though the latter term is somewhat misleading: it provides no redundancy.

Rozdział 20. Localization - I18N/L10N Usage and Setup

Contributed by Andrey Chernov. Rewritten by Michael C. Wu.

20.1. Synopsis

FreeBSD is a very distributed project with users and contributors located all over the world. This chapter discusses the internationalization and localization features of FreeBSD that allow non-English speaking users to get real work done. There are many aspects of the i18n implementation in both the system and application levels, so where applicable we refer the reader to more specific sources of documentation.

After reading this chapter, you will know:

- How different languages and locales are encoded on modern operating systems.
- How to set the locale for your login shell.
- How to configure your console for non-English languages.
- How to use X Window System effectively with different languages.
- Where to find more information about writing i18n-compliant applications.

Before reading this chapter, you should:

- Know how to install additional third-party applications (Rozdział 4).

20.2. The Basics

20.2.1. What Is I18N/L10N?

Developers shortened internationalization into the term I18N, counting the number of letters between the first and the last letters of internationalization. L10N uses the same naming scheme, coming from „localization”. Combined together, I18N/L10N methods, protocols, and applications allow users to use languages of their choice.

I18N applications are programmed using I18N kits under libraries. It allows for developers to write a simple file and translate displayed menus and texts to each language. We strongly encourage programmers to follow this convention.

20.2.2. Why Should I Use I18N/L10N?

I18N/L10N is used whenever you wish to either view, input, or process data in non-English languages.

20.2.3. What Languages Are Supported in the I18N Effort?

I18N and L10N are not FreeBSD specific. Currently, one can choose from most of the major languages of the World, including but not limited to: Chinese, German, Japanese, Korean, French, Russian, Vietnamese and others.

20.3. Using Localization

In all its splendor, I18N is not FreeBSD-specific and is a convention. We encourage you to help FreeBSD in following this convention.

Localization settings are based on three main terms: Language Code, Country Code, and Encoding. Locale names are constructed from these parts as follows:

LanguageCode_CountryCode.Encoding

20.3.1. Language and Country Codes

In order to localize a FreeBSD system to a specific language (or any other I18N-supporting UNIX like systems), the user needs to find out the codes for the specify country and language (country codes tell applications what variation of given language to use). In addition, web browsers, SMTP/POP servers, web servers, etc. make decisions based on them. The following are examples of language/country codes:

Language/Country Code	Description
en_US	English - United States
ru_RU	Russian for Russia
zh_TW	Traditional Chinese for Taiwan

20.3.2. Encodings

Some languages use non-ASCII encodings that are 8-bit, wide or multibyte characters, see `multibyte(3)` for more details. Older applications do not recognize them and mistake them for control characters. Newer applications usually do recognize 8-bit characters. Depending on the implementation, users may be required to compile an application with wide or multibyte characters support, or configure it correctly. To be able to input and process wide or multibyte characters, the FreeBSD Ports Collection (<http://www.FreeBSD.org/ports/index.html>) has provided each language with different programs. Refer to the I18N documentation in the respective FreeBSD Port.

Specifically, the user needs to look at the application documentation to decide on how to configure it correctly or to pass correct values into the `configure/Makefile/compiler`.

Some things to keep in mind are:

- Language specific single C chars character sets (see `multibyte(3)`), e.g. ISO8859-1, ISO8859-15, KOI8-R, CP437.
- Wide or multibyte encodings, e.g. EUC, Big5.

You can check the active list of character sets at the IANA Registry (<http://www.iana.org/assignments/character-sets>).

Notatka: FreeBSD use X11-compatible locale encodings instead.

20.3.3. I18N Applications

In the FreeBSD Ports and Package system, I18N applications have been named with `I18N` in their names for easy identification. However, they do not always support the language needed.

20.3.4. Setting Locale

Usually it is sufficient to export the value of the locale name as `LANG` in the login shell. This could be done in the user's `~/.login_conf` file or in the startup file of the user's shell (`~/.profile`, `~/.bashrc`, `~/.cshrc`). There is no need to set the locale subsets such as `LC_CTYPE`, `LC_TIME`. Please refer to language-specific FreeBSD documentation for more information.

You should set the following two environment variables in your configuration files:

- `LANG` for POSIX `setlocale(3)` family functions
- `MM_CHARSET` for applications' MIME character set

This includes the user shell configuration, the specific application configuration, and the X11 configuration.

20.3.4.1. Setting Locale Methods

There are two methods for setting locale, and both are described below. The first (recommended one) is by assigning the environment variables in login class, and the second is by adding the environment variable assignments to the system's shell startup file.

20.3.4.1.1. Login Classes Method

This method allows environment variables needed for locale name and MIME character sets to be assigned once for every possible shell instead of adding specific shell assignments to each shell's startup file. User Level Setup can be done by an user himself and Administrator Level Setup require superuser privileges.

20.3.4.1.1.1. User Level Setup

Here is a minimal example of a `.login_conf` file in user's home directory which has both variables set for Latin-1 encoding:

```
me:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:
```

Here is an example of a `.login_conf` that sets the variables for Traditional Chinese in BIG-5 encoding. Notice the many more variables set because some software does not respect locale variables correctly for Chinese, Japanese, and Korean.

```
#Users who do not wish to use monetary units or time formats
#of Taiwan can manually change each variable
me:\
    :lang=zh_TW.Big5:\
    :lc_all=zh_TW.Big5:\
    :lc_collate=zh_TW.Big5:\
    :lc_ctype=zh_TW.Big5:\
```

```
:lc_messages=zh_TW.Big5:\
:lc_monetary=zh_TW.Big5:\
:lc_numeric=zh_TW.Big5:\
:lc_time=zh_TW.Big5:\
:charset=big5:\
:xmodifiers="@im=xcin": #Setting the XIM Input Server
```

See Administrator Level Setup and login.conf(5) for more details.

20.3.4.1.1.2. Administrator Level Setup

Verify that the user's login class in `/etc/login.conf` sets the correct language. Make sure these settings appear in `/etc/login.conf`:

```
language_name:accounts_title:\
:charset=MIME_charset:\
:lang=locale_name:\
:tc=default:
```

So sticking with our previous example using Latin-1, it would look like this:

```
german:German Users Accounts:\
:charset=ISO-8859-1:\
:lang=de_DE.ISO8859-1:\
:tc=default:
```

Before changing users Login Classes execute the following command

```
# cap_mkdb /etc/login.conf
```

to make new configuration in `/etc/login.conf` visible to the system.

Changing Login Classes with vipw(8)

Use `vipw` to add new users, and make the entry look like this:

```
user:password:1111:11:language:0:0:User Name:/home/user:/bin/sh
```

Changing Login Classes with adduser(8)

Use `adduser` to add new users, and do the following:

- Set `defaultclass = language` in `/etc/adduser.conf`. Keep in mind you must enter a `default` class for all users of other languages in this case.
- An alternative variant is answering the specified language each time that

```
Enter login class: default []:
```

appears from `adduser(8)`.
- Another alternative is to use the following for each user of a different language that you wish to add:

```
# adduser -class language
```

Changing Login Classes with pw(8)

If you use pw(8) for adding new users, call it in this form:

```
# pw useradd user_name -L language
```

20.3.4.1.2. Shell Startup File Method

Notatka: This method is not recommended because it requires a different setup for each possible shell program chosen. Use the Login Class Method instead.

To add the locale name and MIME character set, just set the two environment variables shown below in the /etc/profile and/or /etc/csh.login shell startup files. We will use the German language as an example below:

In /etc/profile:

```
LANG=de_DE.ISO8859-1; export LANG
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

Or in /etc/csh.login:

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

Alternatively, you can add the above instructions to /usr/share/skel/dot.profile (similar to what was used in /etc/profile above), or /usr/share/skel/dot.login (similar to what was used in /etc/csh.login above).

For X11:

In \$HOME/.xinitrc:

```
LANG=de_DE.ISO8859-1; export LANG
```

Or:

```
setenv LANG de_DE.ISO8859-1
```

Depending on your shell (see above).

20.3.5. Console Setup

For all single C chars character sets, set the correct console fonts in /etc/rc.conf for the language in question with:

```
font8x16=font_name
font8x14=font_name
font8x8=font_name
```

The `font_name` here is taken from the `/usr/share/syscons/fonts` directory, without the `.fnt` suffix.

Also be sure to set the correct keymap and screenmap for your single C chars character set through `sysinstall` (`/stand/sysinstall` in FreeBSD versions older than 5.2). Once inside **sysinstall**, choose **Configure**, then **Console**. Alternatively, you can add the following to `/etc/rc.conf`:

```
scrnmap=screenmap_name
keymap=keymap_name
keychange="fkey_number sequence"
```

The `screenmap_name` here is taken from the `/usr/share/syscons/scrnmaps` directory, without the `.scm` suffix. A screenmap with a corresponding mapped font is usually needed as a workaround for expanding bit 8 to bit 9 on a VGA adapter's font character matrix in pseudographics area, i.e., to move letters out of that area if screen font uses a bit 8 column.

If you have the **moused** daemon enabled by setting the following in your `/etc/rc.conf`:

```
moused_enable="YES"
```

then examine the mouse cursor information in the next paragraph.

By default the mouse cursor of the `syscons(4)` driver occupies the `0xd0-0xd3` range in the character set. If your language uses this range, you need to move the cursor's range outside of it. To enable the workaround for FreeBSD, add the following line to `/etc/rc.conf`:

```
mousechar_start=3
```

The `keymap_name` here is taken from the `/usr/share/syscons/keymaps` directory, without the `.kbd` suffix. If you are uncertain which keymap to use, you can use `kbdmap(1)` to test keymaps without rebooting.

The `keychange` is usually needed to program function keys to match the selected terminal type because function key sequences cannot be defined in the key map.

Also be sure to set the correct console terminal type in `/etc/ttys` for all `ttv*` entries. Current pre-defined correspondences are:

Character Set	Terminal Type
ISO8859-1 or ISO8859-15	cons25l1
ISO8859-2	cons25l2
ISO8859-7	cons25l7
KOI8-R	cons25r
KOI8-U	cons25u
CP437 (VGA default)	cons25
US-ASCII	cons25w

For wide or multibyte characters languages, use the correct FreeBSD port in your `/usr/ports/language` directory. Some ports appear as console while the system sees it as serial vtty's, hence you must reserve enough vtty's for both X11 and the pseudo-serial console. Here is a partial list of applications for using other languages in console:

Language	Location
Traditional Chinese (BIG-5)	chinese/big5con

Language

Japanese
Korean

Location

japanese/kon2-16dot or japanese/mule-freewnn
korean/han

20.3.6. X11 Setup

Although X11 is not part of the FreeBSD Project, we have included some information here for FreeBSD users. For more details, refer to the Xorg web site (<http://www.x.org/>) or whichever X11 Server you use.

In `~/ .Xresources`, you can additionally tune application specific I18N settings (e.g., fonts, menus, etc.).

20.3.6.1. Displaying Fonts

Install **Xorg** server (`x11-servers/xorg-server`) or **XFree86** server (`x11-servers/XFree86-4-Server`), then install the language TrueType fonts. Setting the correct locale should allow you to view your selected language in menus and such.

20.3.6.2. Inputting Non-English Characters

The X11 Input Method (XIM) Protocol is a new standard for all X11 clients. All X11 applications should be written as XIM clients that take input from XIM Input servers. There are several XIM servers available for different languages.

20.3.7. Printer Setup

Some single C chars character sets are usually hardware coded into printers. Wide or multibyte character sets require special setup and we recommend using **apsfilter**. You may also convert the document to PostScript or PDF formats using language specific converters.

20.3.8. Kernel and File Systems

The FreeBSD fast filesystem (FFS) is 8-bit clean, so it can be used with any single C chars character set (see `multibyte(3)`), but there is no character set name stored in the filesystem; i.e., it is raw 8-bit and does not know anything about encoding order. Officially, FFS does not support any form of wide or multibyte character sets yet. However, some wide or multibyte character sets have independent patches for FFS enabling such support. They are only temporary unportable solutions or hacks and we have decided to not include them in the source tree. Refer to respective languages' web sites for more information and the patch files.

The FreeBSD MS-DOS filesystem has the configurable ability to convert between MS-DOS, Unicode character sets and chosen FreeBSD filesystem character sets. See `mount_msdos(8)` for details.

20.4. Compiling I18N Programs

Many FreeBSD Ports have been ported with I18N support. Some of them are marked with -I18N in the port name. These and many other programs have built in support for I18N and need no special consideration.

However, some applications such as **MySQL** need to have the `Makefile` configured with the specific charset. This is usually done in the `Makefile` or done by passing a value to **configure** in the source.

20.5. Localizing FreeBSD to Specific Languages

20.5.1. Russian Language (KOI8-R Encoding)

Originally contributed by Andrey Chernov.

For more information about KOI8-R encoding, see the KOI8-R References (Russian Net Character Set) (<http://koi8.pp.ru/>).

20.5.1.1. Locale Setup

Put the following lines into your `~/ .login_conf` file:

```
me:My Account:\
    :charset=KOI8-R:\
    :lang=ru_RU.KOI8-R:
```

See earlier in this chapter for examples of setting up the locale.

20.5.1.2. Console Setup

- Add the following line to your `/etc/rc.conf` file:

```
mousechar_start=3
```

- Also, use following settings in `/etc/rc.conf`:

```
keymap="ru.koi8-r"
scrnmap="koi8-r2cp866"
font8x16="cp866b-8x16"
font8x14="cp866-8x14"
font8x8="cp866-8x8"
```

- For each `tttyv*` entry in `/etc/ttys`, use `cons25r` as the terminal type.

See earlier in this chapter for examples of setting up the console.

20.5.1.3. Printer Setup

Since most printers with Russian characters come with hardware code page CP866, a special output filter is needed to convert from KOI8-R to CP866. Such a filter is installed by default as `/usr/libexec/lpr/ru/koi2alt`. A Russian printer `/etc/printcap` entry should look like:

```
lp|Russian local line printer:\
    :sh:of=/usr/libexec/lpr/ru/koi2alt:\
    :lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
```

See `printcap(5)` for a detailed description.

20.5.1.4. MS-DOS FS and Russian Filenames

The following example `fstab(5)` entry enables support for Russian filenames in mounted MS-DOS filesystems:

```
/dev/ad0s2      /dos/c  msdos    rw,-Wkoi2dos,-Lru_RU.KOI8-R 0 0
```

The option `-L` selects the locale name used, and `-W` sets the character conversion table. To use the `-w` option, be sure to mount `/usr` before the MS-DOS partition because the conversion tables are located in `/usr/libdata/msdosfs`. For more information, see the `mount_msdos(8)` manual page.

20.5.1.5. X11 Setup

1. Do non-X locale setup first as described.
2. If you use **Xorg**, install `x11-fonts/xorg-fonts-cyrillic` package.

Check the "Files" section in your `/etc/X11/xorg.conf` file. The following lines must be added *before* any other `FontPath` entries:

```
FontPath      "/usr/X11R6/lib/X11/fonts/cyrillic/misc"
FontPath      "/usr/X11R6/lib/X11/fonts/cyrillic/75dpi"
FontPath      "/usr/X11R6/lib/X11/fonts/cyrillic/100dpi"
```

If you use a high resolution video mode, swap the 75 dpi and 100 dpi lines.

3. To activate a Russian keyboard, add the following to the "Keyboard" section of your `xorg.conf` file.

```
Option "XkbLayout"      "us,ru"
Option "XkbOptions"     "grp:toggle"
```

Also make sure that `XkbDisable` is turned off (commented out) there.

For `grp:caps_toggle` the RUS/LAT switch will be **CapsLock**. The old **CapsLock** function is still available via **Shift+CapsLock** (in LAT mode only). For `grp:toggle` the RUS/LAT switch will be **Right Alt**. `grp:caps_toggle` does not work in **Xorg** for unknown reason.

If you have „Windows” keys on your keyboard, and notice that some non-alphabetical keys are mapped incorrectly in RUS mode, add the following line in your `xorg.conf` file.

```
Option "XkbVariant"     ",winkeys"
```

Notatka: The Russian XKB keyboard may not work with non-localized applications.

Notatka: Minimally localized applications should call a `XtSetLanguageProc (NULL, NULL, NULL);` function early in the program.

See KOI8-R for X Window (<http://koi8.pp.ru/xwin.html>) for more instructions on localizing X11 applications.

20.5.2. Traditional Chinese Localization for Taiwan

The FreeBSD-Taiwan Project has an Chinese HOWTO for FreeBSD at <http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/> using many Chinese ports. Current editor for the FreeBSD Chinese HOWTO is Shen Chuan-Hsing <statue@freebsd.sinica.edu.tw>.

Chuan-Hsing Shen <statue@freebsd.sinica.edu.tw> has created the Chinese FreeBSD Collection (CFC) (<http://netlab.cse.yzu.edu.tw/~statue/cfc/>) using FreeBSD-Taiwan's zh-L10N-tut. The packages and the script files are available at <ftp://freebsd.csie.nctu.edu.tw/pub/taiwan/CFC/>.

20.5.3. German Language Localization (for All ISO 8859-1 Languages)

Slaven Rezić <eserte@cs.tu-berlin.de> wrote a tutorial how to use umlauts on a FreeBSD machine. The tutorial is written in German and available at <http://www.de.FreeBSD.org/de/umlaute/>.

20.5.4. Japanese and Korean Language Localization

For Japanese, refer to <http://www.jp.FreeBSD.org/>, and for Korean, refer to <http://www.kr.FreeBSD.org/>.

20.5.5. Non-English FreeBSD Documentation

Some FreeBSD contributors have translated parts of FreeBSD to other languages. They are available through links on the main site (<http://www.FreeBSD.org/index.html>) or in `/usr/share/doc`.

Rozdział 21. The Cutting Edge

Restructured, reorganized, and parts updated by Jim Mock. Original work by Jordan Hubbard, Poul-Henning Kamp, John Polstra, i Nik Clayton.

21.1. Synopsis

FreeBSD is under constant development between releases. For people who want to be on the cutting edge, there are several easy mechanisms for keeping your system in sync with the latest developments. Be warned—the cutting edge is not for everyone! This chapter will help you decide if you want to track the development system, or stick with one of the released versions.

After reading this chapter, you will know:

- The difference between the two development branches: FreeBSD-STABLE and FreeBSD-CURRENT.
- How to keep your system up to date with **CVSup**, **CVS**, or **CTM**.
- How to rebuild and reinstall the entire base system with `make buildworld` (etc).

Before reading this chapter, you should:

- Properly set up your network connection (Rozdział 27).
- Know how to install additional third-party software (Rozdział 4).

21.2. FreeBSD-CURRENT vs. FreeBSD-STABLE

There are two development branches to FreeBSD: FreeBSD-CURRENT and FreeBSD-STABLE. This section will explain a bit about each and describe how to keep your system up-to-date with each respective tree.

FreeBSD-CURRENT will be discussed first, then FreeBSD-STABLE.

21.2.1. Staying Current with FreeBSD

As you read this, keep in mind that FreeBSD-CURRENT is the „bleeding edge” of FreeBSD development. FreeBSD-CURRENT users are expected to have a high degree of technical skill, and should be capable of solving difficult system problems on their own. If you are new to FreeBSD, think twice before installing it.

21.2.1.1. What Is FreeBSD-CURRENT?

FreeBSD-CURRENT is the latest working sources for FreeBSD. This includes work in progress, experimental changes, and transitional mechanisms that might or might not be present in the next official release of the software. While many FreeBSD developers compile the FreeBSD-CURRENT source code daily, there are periods of time when the sources are not buildable. These problems are resolved as expeditiously as possible, but whether or not FreeBSD-CURRENT brings disaster or greatly desired functionality can be a matter of which exact moment you grabbed the source code in!

21.2.1.2. Who Needs FreeBSD-CURRENT?

FreeBSD-CURRENT is made available for 3 primary interest groups:

1. Members of the FreeBSD community who are actively working on some part of the source tree and for whom keeping „current” is an absolute requirement.
2. Members of the FreeBSD community who are active testers, willing to spend time solving problems in order to ensure that FreeBSD-CURRENT remains as sane as possible. These are also people who wish to make topical suggestions on changes and the general direction of FreeBSD, and submit patches to implement them.
3. Those who merely wish to keep an eye on things, or to use the current sources for reference purposes (e.g. for *reading*, not running). These people also make the occasional comment or contribute code.

21.2.1.3. What Is FreeBSD-CURRENT Not?

1. A fast-track to getting pre-release bits because you heard there is some cool new feature in there and you want to be the first on your block to have it. Being the first on the block to get the new feature means that you are the first on the block to get the new bugs.
2. A quick way of getting bug fixes. Any given version of FreeBSD-CURRENT is just as likely to introduce new bugs as to fix existing ones.
3. In any way „officially supported”. We do our best to help people genuinely in one of the 3 „legitimate” FreeBSD-CURRENT groups, but we simply *do not have the time* to provide tech support. This is not because we are mean and nasty people who do not like helping people out (we would not even be doing FreeBSD if we were). We simply cannot answer hundreds messages a day *and* work on FreeBSD! Given the choice between improving FreeBSD and answering lots of questions on experimental code, the developers opt for the former.

21.2.1.4. Using FreeBSD-CURRENT

1. Join the freebsd-current (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) and the cvs-all (<http://lists.FreeBSD.org/mailman/listinfo/cvs-all>) lists. This is not just a good idea, it is *essential*. If you are not on the *freebsd-current* (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) list, you will not see the comments that people are making about the current state of the system and thus will probably end up stumbling over a lot of problems that others have already found and solved. Even more importantly, you will miss out on important bulletins which may be critical to your system’s continued health.

The cvs-all (<http://lists.FreeBSD.org/mailman/listinfo/cvs-all>) list will allow you to see the commit log entry for each change as it is made along with any pertinent information on possible side-effects.

To join these lists, or one of the others available go to <http://lists.FreeBSD.org/mailman/listinfo> and click on the list that you wish to subscribe to. Instructions on the rest of the procedure are available there.

2. Grab the sources from a FreeBSD mirror site. You can do this in one of two ways:
 - a. Use the cvsup program with the supfile named `standard-supfile` available from `/usr/share/examples/cvsup`. This is the most recommended method, since it allows you to grab the entire collection once and then only what has changed from then on. Many people run `cvsup` from `cron` and keep their sources up-to-date automatically. You have to customize the sample `supfile` above, and configure `cvsup` for your environment.

- b. Use the **CTM** facility. If you have very bad connectivity (high price connections or only email access) **CTM** is an option. However, it is a lot of hassle and can give you broken files. This leads to it being rarely used, which again increases the chance of it not working for fairly long periods of time. We recommend using **CVSup** for anybody with a 9600 bps modem or faster connection.
3. If you are grabbing the sources to run, and not just look at, then grab *all* of FreeBSD-CURRENT, not just selected portions. The reason for this is that various parts of the source depend on updates elsewhere, and trying to compile just a subset is almost guaranteed to get you into trouble.
- Before compiling FreeBSD-CURRENT, read the Makefile in `/usr/src` carefully. You should at least install a new kernel and rebuild the world the first time through as part of the upgrading process. Reading the FreeBSD-CURRENT mailing list (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) and `/usr/src/UPDATING` will keep you up-to-date on other bootstrapping procedures that sometimes become necessary as we move toward the next release.
4. Be active! If you are running FreeBSD-CURRENT, we want to know what you have to say about it, especially if you have suggestions for enhancements or bug fixes. Suggestions with accompanying code are received most enthusiastically!

21.2.2. Staying Stable with FreeBSD

21.2.2.1. What Is FreeBSD-STABLE?

FreeBSD-STABLE is our development branch from which major releases are made. Changes go into this branch at a different pace, and with the general assumption that they have first gone into FreeBSD-CURRENT for testing. This is *still* a development branch, however, and this means that at any given time, the sources for FreeBSD-STABLE may or may not be suitable for any particular purpose. It is simply another engineering development track, not a resource for end-users.

21.2.2.2. Who Needs FreeBSD-STABLE?

If you are interested in tracking or contributing to the FreeBSD development process, especially as it relates to the next „point” release of FreeBSD, then you should consider following FreeBSD-STABLE.

While it is true that security fixes also go into the FreeBSD-STABLE branch, you do not *need* to track FreeBSD-STABLE to do this. Every security advisory for FreeBSD explains how to fix the problem for the releases it affects ¹, and tracking an entire development branch just for security reasons is likely to bring in a lot of unwanted changes as well.

Although we endeavor to ensure that the FreeBSD-STABLE branch compiles and runs at all times, this cannot be guaranteed. In addition, while code is developed in FreeBSD-CURRENT before including it in FreeBSD-STABLE, more people run FreeBSD-STABLE than FreeBSD-CURRENT, so it is inevitable that bugs and corner cases will sometimes be found in FreeBSD-STABLE that were not apparent in FreeBSD-CURRENT.

For these reasons, we do *not* recommend that you blindly track FreeBSD-STABLE, and it is particularly important that you do not update any production servers to FreeBSD-STABLE without first thoroughly testing the code in your development environment.

If you do not have the resources to do this then we recommend that you run the most recent release of FreeBSD, and use the binary update mechanism to move from release to release.

21.2.2.3. Using FreeBSD-STABLE

1. Join the `freebsd-stable` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>) list. This will keep you informed of build-dependencies that may appear in FreeBSD-STABLE or any other issues requiring special attention. Developers will also make announcements in this mailing list when they are contemplating some controversial fix or update, giving the users a chance to respond if they have any issues to raise concerning the proposed change.

The `cvs-all` (<http://lists.FreeBSD.org/mailman/listinfo/cvs-all>) list will allow you to see the commit log entry for each change as it is made along with any pertinent information on possible side-effects.

To join these lists, or one of the others available go to <http://lists.FreeBSD.org/mailman/listinfo> and click on the list that you wish to subscribe to. Instructions on the rest of the procedure are available there.

2. If you are going to install a new system and want it to run monthly snapshot built from FreeBSD-STABLE, please check the `Snapshots` (<http://www.FreeBSD.org/snapshots/>) web page for more information. Alternatively, it is possible to install the most recent FreeBSD-STABLE release from the mirror sites and follow the instructions below to upgrade your system to the most up to date FreeBSD-STABLE source code.

If you are already running a previous release of FreeBSD and wish to upgrade via sources then you can easily do so from FreeBSD mirror site. This can be done in one of two ways:

- a. Use the `cvsup` program with the `supfile` named `stable-supfile` from the directory `/usr/share/examples/cvsup`. This is the most recommended method, since it allows you to grab the entire collection once and then only what has changed from then on. Many people run `cvsup` from `cron` to keep their sources up-to-date automatically. You have to customize the sample `supfile` above, and configure `cvsup` for your environment.
 - b. Use the **CTM** facility. If you do not have a fast and inexpensive connection to the Internet, this is the method you should consider using.
3. Essentially, if you need rapid on-demand access to the source and communications bandwidth is not a consideration, use `cvsup` or `ftp`. Otherwise, use **CTM**.
 4. Before compiling FreeBSD-STABLE, read the `Makefile` in `/usr/src` carefully. You should at least install a new kernel and rebuild the world the first time through as part of the upgrading process. Reading the FreeBSD-STABLE mailing list (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>) and `/usr/src/UPDATING` will keep you up-to-date on other bootstrapping procedures that sometimes become necessary as we move toward the next release.

21.3. Synchronizing Your Source

There are various ways of using an Internet (or email) connection to stay up-to-date with any given area of the FreeBSD project sources, or all areas, depending on what interests you. The primary services we offer are Anonymous CVS, CVSup, and CTM.

Ostrzeżenie While it is possible to update only parts of your source tree, the only supported update procedure is to update the entire tree and recompile both userland (i.e., all the programs that run in user space, such as those in `/bin` and `/sbin`) and kernel sources. Updating only part of your source tree, only the kernel, or only userland will often result in problems. These problems may range from compile errors to kernel panics or data corruption.

Anonymous CVS and **CVSup** use the *pull* model of updating sources. In the case of **CVSup** the user (or a `cron` script) invokes the `cvsup` program, and it interacts with a `cvsupd` server somewhere to bring your files up-to-date. The updates you receive are up-to-the-minute and you get them when, and only when, you want them. You can easily restrict your updates to the specific files or directories that are of interest to you. Updates are generated on the fly by the server, according to what you have and what you want to have. **Anonymous CVS** is quite a bit more simplistic than **CVSup** in that it is just an extension to **CVS** which allows it to pull changes directly from a remote CVS repository. **CVSup** can do this far more efficiently, but **Anonymous CVS** is easier to use.

CTM, on the other hand, does not interactively compare the sources you have with those on the master archive or otherwise pull them across. Instead, a script which identifies changes in files since its previous run is executed several times a day on the master CTM machine, any detected changes being compressed, stamped with a sequence-number and encoded for transmission over email (in printable ASCII only). Once received, these „CTM deltas” can then be handed to the `ctm_rmail(1)` utility which will automatically decode, verify and apply the changes to the user’s copy of the sources. This process is far more efficient than **CVSup**, and places less strain on our server resources since it is a *push* rather than a *pull* model.

There are other trade-offs, of course. If you inadvertently wipe out portions of your archive, **CVSup** will detect and rebuild the damaged portions for you. **CTM** will not do this, and if you wipe some portion of your source tree out (and do not have it backed up) then you will have to start from scratch (from the most recent CVS „base delta”) and rebuild it all with **CTM** or, with **Anonymous CVS**, simply delete the bad bits and resync.

21.4. Rebuilding „world”

Once you have synchronized your local source tree against a particular version of FreeBSD (FreeBSD-STABLE, FreeBSD-CURRENT, and so on) you can then use the source tree to rebuild the system.

Make a Backup It cannot be stressed enough how important it is to make a backup of your system *before* you do this. While rebuilding the world is (as long as you follow these instructions) an easy task to do, there will inevitably be times when you make mistakes, or when mistakes made by others in the source tree render your system unbootable.

Make sure you have taken a backup. And have a fixit floppy or bootable CD at hand. You will probably never have to use it, but it is better to be safe than sorry!

Subscribe to the Right Mailing List The FreeBSD-STABLE and FreeBSD-CURRENT branches are, by their nature, *in development*. People that contribute to FreeBSD are human, and mistakes occasionally happen.

Sometimes these mistakes can be quite harmless, just causing your system to print a new diagnostic warning. Or the change may be catastrophic, and render your system unbootable or destroy your file systems (or worse).

If problems like these occur, a „heads up” is posted to the appropriate mailing list, explaining the nature of the problem and which systems it affects. And an „all clear” announcement is posted when the problem has been solved.

If you try to track FreeBSD-STABLE or FreeBSD-CURRENT and do not read the FreeBSD-STABLE mailing list (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>) or the FreeBSD-CURRENT mailing list (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>) respectively, then you are asking for trouble.

Do not use `make world`A lot of older documentation recommends using `make world` for this. Doing that skips some important steps and should only be used if you are sure of what you are doing. For almost all circumstances `make world` is the wrong thing to do, and the procedure described here should be used instead.

21.4.1. The Canonical Way to Update Your System

To update your system, you should check `/usr/src/UPDATING` for any pre-buildworld steps necessary for your version of the sources and then use the following procedure:

```
# make buildworld
# make buildkernel
# make installkernel
# reboot
```

Notatka: There are a few rare cases when an extra run of `mergemaster -p` is needed before the `buildworld` step. These are described in `UPDATING`. In general, though, you can safely omit this step if you are not updating across one or more major FreeBSD versions.

After `installkernel` finishes successfully, you should boot in single user mode (i.e. using `boot -s` from the loader prompt). Then run:

```
# mergemaster -p
# make installworld
# mergemaster
# reboot
```

Read Further ExplanationsThe sequence described above is only a short resume to help you getting started. You should however read the following sections to clearly understand each step, especially if you want to use a custom kernel configuration.

21.4.2. Read `/usr/src/UPDATING`

Before you do anything else, read `/usr/src/UPDATING` (or the equivalent file wherever you have a copy of the source code). This file should contain important information about problems you might encounter, or specify the order in which you might have to run certain commands. If `UPDATING` contradicts something you read here, `UPDATING` takes precedence.

WAŻNE: Reading `UPDATING` is not an acceptable substitute for subscribing to the correct mailing list, as described previously. The two requirements are complementary, not exclusive.

21.4.3. Check `/etc/make.conf`

Examine the files `/usr/share/examples/etc/make.conf` and `/etc/make.conf`. The first contains some default defines – most of which are commented out. To make use of them when you rebuild your system from source, add them to `/etc/make.conf`. Keep in mind that anything you add to `/etc/make.conf` is also used every time you run `make`, so it is a good idea to set them to something sensible for your system.

A typical user will probably want to copy the `CFLAGS` and `NO_PROFILE` lines found in `/usr/share/examples/etc/make.conf` to `/etc/make.conf` and uncomment them.

Examine the other definitions (`COPTFLAGS`, `NOPORTDOCS` and so on) and decide if they are relevant to you.

21.4.4. Update the Files in `/etc`

The `/etc` directory contains a large part of your system's configuration information, as well as scripts that are run at system startup. Some of these scripts change from version to version of FreeBSD.

Some of the configuration files are also used in the day to day running of the system. In particular, `/etc/group`.

There have been occasions when the installation part of `make installworld` has expected certain usernames or groups to exist. When performing an upgrade it is likely that these users or groups did not exist. This caused problems when upgrading. In some cases `make buildworld` will check to see if these users or groups exist.

An example of this is when the `smmsp` user was added. Users had the installation process fail for them when `mtree(8)` was trying to create `/var/spool/clientmqueue`.

The solution is to run `mergemaster(8)` in pre-buildworld mode by providing the `-p` option. This will compare only those files that are essential for the success of `buildworld` or `installworld`. If your old version of `mergemaster` does not support `-p`, use the new version in the source tree when running for the first time:

```
# cd /usr/src/usr.sbin/mergemaster
# ./mergemaster.sh -p
```

Podpowiedź: If you are feeling particularly paranoid, you can check your system to see which files are owned by the group you are renaming or deleting:

```
# find / -group GID -print
```

will show all files owned by group `GID` (which can be either a group name or a numeric group ID).

21.4.5. Drop to Single User Mode

You may want to compile the system in single user mode. Apart from the obvious benefit of making things go slightly faster, reinstalling the system will touch a lot of important system files, all the standard system binaries,

libraries, include files and so on. Changing these on a running system (particularly if you have active users on the system at the time) is asking for trouble.

Another method is to compile the system in multi-user mode, and then drop into single user mode for the installation. If you would like to do it this way, simply hold off on the following steps until the build has completed. You can postpone dropping to single user mode until you have to `installkernel` or `installworld`.

As the superuser, you can execute:

```
# shutdown now
```

from a running system, which will drop it to single user mode.

Alternatively, reboot the system, and at the boot prompt, select the „single user” option. The system will then boot single user. At the shell prompt you should then run:

```
# fsck -p
# mount -u /
# mount -a -t ufs
# swapon -a
```

This checks the file systems, remounts / read/write, mounts all the other UFS file systems referenced in `/etc/fstab` and then turns swapping on.

Notatka: If your CMOS clock is set to local time and not to GMT (this is true if the output of the `date(1)` command does not show the correct time and zone), you may also need to run the following command:

```
# adjkerntz -i
```

This will make sure that your local time-zone settings get set up correctly — without this, you may later run into some problems.

21.4.6. Remove `/usr/obj`

As parts of the system are rebuilt they are placed in directories which (by default) go under `/usr/obj`. The directories shadow those under `/usr/src`.

You can speed up the `make buildworld` process, and possibly save yourself some dependency headaches by removing this directory as well.

Some files below `/usr/obj` may have the immutable flag set (see `chflags(1)` for more information) which must be removed first.

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

21.4.7. Recompile the Base System

21.4.7.1. Saving the Output

It is a good idea to save the output you get from running `make(1)` to another file. If something goes wrong you will have a copy of the error message. While this might not help you in diagnosing what has gone wrong, it can help others if you post your problem to one of the FreeBSD mailing lists.

The easiest way to do this is to use the `script(1)` command, with a parameter that specifies the name of the file to save all output to. You would do this immediately before rebuilding the world, and then type `exit` when the process has finished.

```
# script /var/tmp/mw.out
Script started, output file is /var/tmp/mw.out
# make TARGET
... compile, compile, compile ...
# exit
Script done, ...
```

If you do this, *do not* save the output in `/tmp`. This directory may be cleared next time you reboot. A better place to store it is in `/var/tmp` (as in the previous example) or in `root`'s home directory.

21.4.7.2. Compile the Base System

You must be in the `/usr/src` directory:

```
# cd /usr/src
```

(unless, of course, your source code is elsewhere, in which case change to that directory instead).

To rebuild the world you use the `make(1)` command. This command reads instructions from the `Makefile`, which describes how the programs that comprise FreeBSD should be rebuilt, the order in which they should be built, and so on.

The general format of the command line you will type is as follows:

```
# make -x -DVARIABLE target
```

In this example, `-x` is an option that you would pass to `make(1)`. See the `make(1)` manual page for an example of the options you can pass.

`-DVARIABLE` passes a variable to the `Makefile`. The behavior of the `Makefile` is controlled by these variables. These are the same variables as are set in `/etc/make.conf`, and this provides another way of setting them.

```
# make -DNO_PROFILE target
```

is another way of specifying that profiled libraries should not be built, and corresponds with the

```
NO_PROFILE=    true    #    Avoid compiling profiled libraries
```

line in `/etc/make.conf`.

`target` tells `make(1)` what you want to do. Each `Makefile` defines a number of different „targets”, and your choice of target determines what happens.

Some targets are listed in the `Makefile`, but are not meant for you to run. Instead, they are used by the build process to break out the steps necessary to rebuild the system into a number of sub-steps.

Most of the time you will not need to pass any parameters to `make(1)`, and so your command line will look like this:

```
# make target
```

Where `target` will be one of many build options. The first target should always be `buildworld`.

As the names imply, `buildworld` builds a complete new tree under `/usr/obj`, and `installworld`, another target, installs this tree on the current machine.

Having separate options is very useful for two reasons. First, it allows you to do the build safe in the knowledge that no components of your running system will be affected. The build is „self hosted”. Because of this, you can safely run `buildworld` on a machine running in multi-user mode with no fear of ill-effects. It is still recommended that you run the `installworld` part in single user mode, though.

Secondly, it allows you to use NFS mounts to upgrade multiple machines on your network. If you have three machines, A, B and C that you want to upgrade, run `make buildworld` and `make installworld` on A. B and C should then NFS mount `/usr/src` and `/usr/obj` from A, and you can then run `make installworld` to install the results of the build on B and C.

Although the `world` target still exists, you are strongly encouraged not to use it.

Run

```
# make buildworld
```

It is possible to specify a `-j` option to `make` which will cause it to spawn several simultaneous processes. This is most useful on multi-CPU machines. However, since much of the compiling process is IO bound rather than CPU bound it is also useful on single CPU machines.

On a typical single-CPU machine you would run:

```
# make -j4 buildworld
```

`make(1)` will then have up to 4 processes running at any one time. Empirical evidence posted to the mailing lists shows this generally gives the best performance benefit.

If you have a multi-CPU machine and you are using an SMP configured kernel try values between 6 and 10 and see how they speed things up.

21.4.7.3. Timings

Many factors influence the build time, but fairly recent machines may only take a one or two hours to build the FreeBSD-STABLE tree, with no tricks or shortcuts used during the process. A FreeBSD-CURRENT tree will take somewhat longer.

21.4.8. Compile and Install a New Kernel

To take full advantage of your new system you should recompile the kernel. This is practically a necessity, as certain memory structures may have changed, and programs like `ps(1)` and `top(1)` will fail to work until the kernel and source code versions are the same.

The simplest, safest way to do this is to build and install a kernel based on `GENERIC`. While `GENERIC` may not have all the necessary devices for your system, it should contain everything necessary to boot your system back to single user mode. This is a good test that the new system works properly. After booting from `GENERIC` and verifying that your system works you can then build a new kernel based on your normal kernel configuration file.

On FreeBSD it is important to build world before building a new kernel.

Notatka: If you want to build a custom kernel, and already have a configuration file, just use `KERNCONF=MYKERNEL` like this:

```
# cd /usr/src
# make buildkernel KERNCONF=MYKERNEL
# make installkernel KERNCONF=MYKERNEL
```

Note that if you have raised `kern.securelevel` above 1 *and* you have set either the `noschg` or similar flags to your kernel binary, you might find it necessary to drop into single user mode to use `installkernel`. Otherwise you should be able to run both these commands from multi user mode without problems. See `init(8)` for details about `kern.securelevel` and `chflags(1)` for details about the various file flags.

21.4.9. Reboot into Single User Mode

You should reboot into single user mode to test the new kernel works. Do this by following the instructions in Sekcja 21.4.5.

21.4.10. Install the New System Binaries

If you were building a version of FreeBSD recent enough to have used `make buildworld` then you should now use `installworld` to install the new system binaries.

Run

```
# cd /usr/src
# make installworld
```

Notatka: If you specified variables on the `make buildworld` command line, you must specify the same variables in the `make installworld` command line. This does not necessarily hold true for other options; for example, `-j` must never be used with `installworld`.

For example, if you ran:

```
# make -DNO_PROFILE buildworld
```

you must install the results with:

```
# make -DNO_PROFILE installworld
```

otherwise it would try to install profiled libraries that had not been built during the `make buildworld` phase.

21.4.11. Update Files Not Updated by `make installworld`

Remaking the world will not update certain directories (in particular, `/etc`, `/var` and `/usr`) with new or changed configuration files.

The simplest way to update these files is to use `mergemaster(8)`, though it is possible to do it manually if you would prefer to do that. Regardless of which way you choose, be sure to make a backup of `/etc` in case anything goes wrong.

21.4.11.1. `mergemaster`

Contributed by Tom Rhodes.

The `mergemaster(8)` utility is a Bourne script that will aid you in determining the differences between your configuration files in `/etc`, and the configuration files in the source tree `/usr/src/etc`. This is the recommended solution for keeping the system configuration files up to date with those located in the source tree.

To begin simply type `mergemaster` at your prompt, and watch it start going. `mergemaster` will then build a temporary root environment, from `/` down, and populate it with various system configuration files. Those files are then compared to the ones currently installed in your system. At this point, files that differ will be shown in `diff(1)` format, with the `+` sign representing added or modified lines, and `-` representing lines that will be either removed completely, or replaced with a new line. See the `diff(1)` manual page for more information about the `diff(1)` syntax and how file differences are shown.

`mergemaster(8)` will then show you each file that displays variances, and at this point you will have the option of either deleting the new file (referred to as the temporary file), installing the temporary file in its unmodified state, merging the temporary file with the currently installed file, or viewing the `diff(1)` results again.

Choosing to delete the temporary file will tell `mergemaster(8)` that we wish to keep our current file unchanged, and to delete the new version. This option is not recommended, unless you see no reason to change the current file. You can get help at any time by typing `?` at the `mergemaster(8)` prompt. If the user chooses to skip a file, it will be presented again after all other files have been dealt with.

Choosing to install the unmodified temporary file will replace the current file with the new one. For most unmodified files, this is the best option.

Choosing to merge the file will present you with a text editor, and the contents of both files. You can now merge them by reviewing both files side by side on the screen, and choosing parts from both to create a finished product. When the files are compared side by side, the `l` key will select the left contents and the `r` key will select contents from your right. The final output will be a file consisting of both parts, which can then be installed. This option is customarily used for files where settings have been modified by the user.

Choosing to view the `diff(1)` results again will show you the file differences just like `mergemaster(8)` did before prompting you for an option.

After `mergemaster(8)` is done with the system files you will be prompted for other options. `mergemaster(8)` may ask if you want to rebuild the password file and will finish up with an option to remove left-over temporary files.

21.4.11.2. Manual Update

If you wish to do the update manually, however, you cannot just copy over the files from `/usr/src/etc` to `/etc` and have it work. Some of these files must be „installed” first. This is because the `/usr/src/etc` directory *is not* a copy of what your `/etc` directory should look like. In addition, there are files that should be in `/etc` that are not in `/usr/src/etc`.

If you are using mergemaster(8) (as recommended), you can skip forward to the next section.

The simplest way to do this by hand is to install the files into a new directory, and then work through them looking for differences.

Backup Your Existing /etc Although, in theory, nothing is going to touch this directory automatically, it is always better to be sure. So copy your existing /etc directory somewhere safe. Something like:

```
# cp -Rp /etc /etc.old
```

-R does a recursive copy, -p preserves times, ownerships on files and suchlike.

You need to build a dummy set of directories to install the new /etc and other files into. /var/tmp/root is a reasonable choice, and there are a number of subdirectories required under this as well.

```
# mkdir /var/tmp/root
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root distrib-dirs distribution
```

This will build the necessary directory structure and install the files. A lot of the subdirectories that have been created under /var/tmp/root are empty and should be deleted. The simplest way to do this is to:

```
# cd /var/tmp/root
# find -d . -type d | xargs rmdir 2>/dev/null
```

This will remove all empty directories. (Standard error is redirected to /dev/null to prevent the warnings about the directories that are not empty.)

/var/tmp/root now contains all the files that should be placed in appropriate locations below /. You now have to go through each of these files, determining how they differ with your existing files.

Note that some of the files that will have been installed in /var/tmp/root have a leading „,“. At the time of writing the only files like this are shell startup files in /var/tmp/root/ and /var/tmp/root/root/, although there may be others (depending on when you are reading this). Make sure you use `ls -a` to catch them.

The simplest way to do this is to use `diff(1)` to compare the two files:

```
# diff /etc/shells /var/tmp/root/etc/shells
```

This will show you the differences between your /etc/shells file and the new /var/tmp/root/etc/shells file. Use these to decide whether to merge in changes that you have made or whether to copy over your old file.

Name the New Root Directory (/var/tmp/root) with a Time Stamp, so You Can Easily Compare

Differences Between Versions: Frequently rebuilding the world means that you have to update /etc frequently as well, which can be a bit of a chore.

You can speed this process up by keeping a copy of the last set of changed files that you merged into /etc. The following procedure gives one idea of how to do this.

1. Make the world as normal. When you want to update /etc and the other directories, give the target directory a name based on the current date. If you were doing this on the 14th of February 1998 you could do the following:

```
# mkdir /var/tmp/root-19980214
```

```
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root-19980214 \
  distrib-dirs distribution
```

2. Merge in the changes from this directory as outlined above.

Do not remove the `/var/tmp/root-19980214` directory when you have finished.

3. When you have downloaded the latest version of the source and remade it, follow step 1. This will give you a new directory, which might be called `/var/tmp/root-19980221` (if you wait a week between doing updates).
4. You can now see the differences that have been made in the intervening week using `diff(1)` to create a recursive diff between the two directories:

```
# cd /var/tmp
# diff -r root-19980214 root-19980221
```

Typically, this will be a much smaller set of differences than those between `/var/tmp/root-19980221/etc` and `/etc`. Because the set of differences is smaller, it is easier to migrate those changes across into your `/etc` directory.

5. You can now remove the older of the two `/var/tmp/root-*` directories:

```
# rm -rf /var/tmp/root-19980214
```

6. Repeat this process every time you need to merge in changes to `/etc`.

You can use `date(1)` to automate the generation of the directory names:

```
# mkdir /var/tmp/root-`date "+%Y%m%d"`
```

21.4.12. Rebooting

You are now done. After you have verified that everything appears to be in the right place you can reboot the system. A simple `shutdown(8)` should do it:

```
# shutdown -r now
```

21.4.13. Finished

You should now have successfully upgraded your FreeBSD system. Congratulations.

If things went slightly wrong, it is easy to rebuild a particular piece of the system. For example, if you accidentally deleted `/etc/magic` as part of the upgrade or merge of `/etc`, the `file(1)` command will stop working. In this case, the fix would be to run:

```
# cd /usr/src/usr.bin/file
# make all install
```

21.4.14. Questions

1. Do I need to re-make the world for every change?

There is no easy answer to this one, as it depends on the nature of the change. For example, if you just ran **CVSup**, and it has shown the following files as being updated:

```
src/games/cribbage/instr.c
src/games/sail/pl_main.c
src/release/sysinstall/config.c
src/release/sysinstall/media.c
src/share/mk/bsd.port.mk
```

it probably is not worth rebuilding the entire world. You could just go to the appropriate sub-directories and make `all install`, and that's about it. But if something major changed, for example `src/lib/libc/stdlib` then you should either re-make the world, or at least those parts of it that are statically linked (as well as anything else you might have added that is statically linked).

At the end of the day, it is your call. You might be happy re-making the world every fortnight say, and let changes accumulate over that fortnight. Or you might want to re-make just those things that have changed, and be confident you can spot all the dependencies.

And, of course, this all depends on how often you want to upgrade, and whether you are tracking FreeBSD-STABLE or FreeBSD-CURRENT.

2. My compile failed with lots of signal 11 (or other signal number) errors. What has happened?

This is normally indicative of hardware problems. (Re)making the world is an effective way to stress test your hardware, and will frequently throw up memory problems. These normally manifest themselves as the compiler mysteriously dying on receipt of strange signals.

A sure indicator of this is if you can restart the make and it dies at a different point in the process.

In this instance there is little you can do except start swapping around the components in your machine to determine which one is failing.

3. Can I remove `/usr/obj` when I have finished?

The short answer is yes.

`/usr/obj` contains all the object files that were produced during the compilation phase. Normally, one of the first steps in the `make buildworld` process is to remove this directory and start afresh. In this case, keeping `/usr/obj` around after you have finished makes little sense, and will free up a large chunk of disk space (currently about 340 MB).

However, if you know what you are doing you can have `make buildworld` skip this step. This will make subsequent builds run much faster, since most of sources will not need to be recompiled. The flip side of this is that subtle dependency problems can creep in, causing your build to fail in odd ways. This frequently generates noise on the FreeBSD mailing lists, when one person complains that their build has failed, not realizing that it is because they have tried to cut corners.

4. Can interrupted builds be resumed?

This depends on how far through the process you got before you found a problem.

In general (and this is not a hard and fast rule) the `make buildworld` process builds new copies of essential tools (such as `gcc(1)`, and `make(1)`) and the system libraries. These tools and libraries are then installed. The new tools and libraries are then used to rebuild themselves, and are installed again. The entire system (now including regular user programs, such as `ls(1)` or `grep(1)`) is then rebuilt with the new system files.

If you are at the last stage, and you know it (because you have looked through the output that you were storing) then you can (fairly safely) do:

```
... fix the problem ...
# cd /usr/src
# make -DNO_CLEAN all
```

This will not undo the work of the previous `make buildworld`.

If you see the message:

```
-----
Building everything..
-----
```

in the `make buildworld` output then it is probably fairly safe to do so.

If you do not see that message, or you are not sure, then it is always better to be safe than sorry, and restart the build from scratch.

5. How can I speed up making the world?

- Run in single user mode.
- Put the `/usr/src` and `/usr/obj` directories on separate file systems held on separate disks. If possible, put these disks on separate disk controllers.
- Better still, put these file systems across multiple disks using the `ccd(4)` (concatenated disk driver) device.
- Turn off profiling (set „`NO_PROFILE=true`” in `/etc/make.conf`). You almost certainly do not need it.
- Also in `/etc/make.conf`, set `CFLAGS` to something like `-O -pipe`. The optimization `-O2` is much slower, and the optimization difference between `-O` and `-O2` is normally negligible. `-pipe` lets the compiler use pipes rather than temporary files for communication, which saves disk access (at the expense of memory).
- Pass the `-jn` option to `make(1)` to run multiple processes in parallel. This usually helps regardless of whether you have a single or a multi processor machine.
- The file system holding `/usr/src` can be mounted (or remounted) with the `noatime` option. This prevents the file system from recording the file access time. You probably do not need this information anyway.

```
# mount -u -o noatime /usr/src
```

Ostrzeżenie The example assumes `/usr/src` is on its own file system. If it is not (if it is a part of `/usr` for example) then you will need to use that file system mount point, and not `/usr/src`.

- The file system holding `/usr/obj` can be mounted (or remounted) with the `async` option. This causes disk writes to happen asynchronously. In other words, the write completes immediately, and the data is written to the disk a few seconds later. This allows writes to be clustered together, and can be a dramatic performance boost.

Ostrzeżenie Keep in mind that this option makes your file system more fragile. With this option there is an increased chance that, should power fail, the file system will be in an unrecoverable state when the machine restarts.

If `/usr/obj` is the only thing on this file system then it is not a problem. If you have other, valuable data on the same file system then ensure your backups are fresh before you enable this option.

```
# mount -u -o async /usr/obj
```

Ostrzeżenie As above, if `/usr/obj` is not on its own file system, replace it in the example with the name of the appropriate mount point.

6. What do I do if something goes wrong?

Make absolutely sure your environment has no extraneous cruft from earlier builds. This is simple enough.

```
# chflags -R noschg /usr/obj/usr
# rm -rf /usr/obj/usr
# cd /usr/src
# make cleandir
# make cleandir
```

Yes, `make cleandir` really should be run twice.

Then restart the whole process, starting with `make buildworld`.

If you still have problems, send the error and the output of `uname -a` to Ogólna lista dyskusyjna FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>). Be prepared to answer other questions about your setup!

21.5. Tracking for Multiple Machines

Contributed by Mike Meyer.

If you have multiple machines that you want to track the same source tree, then having all of them download sources and rebuild everything seems like a waste of resources: disk space, network bandwidth, and CPU cycles. It is, and the solution is to have one machine do most of the work, while the rest of the machines mount that work via NFS. This section outlines a method of doing so.

21.5.1. Preliminaries

First, identify a set of machines that is going to run the same set of binaries, which we will call a *build set*. Each machine can have a custom kernel, but they will be running the same userland binaries. From that set, choose a machine to be the *build machine*. It is going to be the machine that the world and kernel are built on. Ideally, it should be a fast machine that has sufficient spare CPU to run `make buildworld` and `make buildkernel`. You will also want to choose a machine to be the *test machine*, which will test software updates before they are put into production. This *must* be a machine that you can afford to have down for an extended period of time. It can be the build machine, but need not be.

All the machines in this build set need to mount `/usr/obj` and `/usr/src` from the same machine, and at the same point. Ideally, those are on two different drives on the build machine, but they can be NFS mounted on that machine as well. If you have multiple build sets, `/usr/src` should be on one build machine, and NFS mounted on the rest.

Finally make sure that `/etc/make.conf` on all the machines in the build set agrees with the build machine. That means that the build machine must build all the parts of the base system that any machine in the build set is going to install. Also, each build machine should have its kernel name set with `KERNCONF` in `/etc/make.conf`, and the build machine should list them all in `KERNCONF`, listing its own kernel first. The build machine must have the kernel configuration files for each machine in `/usr/src/sys/arch/conf` if it is going to build their kernels.

21.5.2. The Base System

Now that all that is done, you are ready to build everything. Build the kernel and world as described in Sekcja 21.4.7.2 on the build machine, but do not install anything. After the build has finished, go to the test machine, and install the kernel you just built. If this machine mounts `/usr/src` and `/usr/obj` via NFS, when you reboot to single user you will need to enable the network and mount them. The easiest way to do this is to boot to multi-user, then run `shutdown now` to go to single user mode. Once there, you can install the new kernel and world and run `mergemaster` just as you normally would. When done, reboot to return to normal multi-user operations for this machine.

After you are certain that everything on the test machine is working properly, use the same procedure to install the new software on each of the other machines in the build set.

21.5.3. Ports

The same ideas can be used for the ports tree. The first critical step is mounting `/usr/ports` from the same machine to all the machines in the build set. You can then set up `/etc/make.conf` properly to share distfiles. You should set `DISTDIR` to a common shared directory that is writable by whichever user `root` is mapped to by your NFS mounts. Each machine should set `WRKDIRPREFIX` to a local build directory. Finally, if you are going to be building and distributing packages, you should set `PACKAGES` to a directory similar to `DISTDIR`.

Przypisy

1. That is not quite true. We can not continue to support old releases of FreeBSD forever, although we do support them for many years. For a complete description of the current security policy for old releases of FreeBSD, please see <http://www.FreeBSD.org/security/>.

IV. Komunikacja sieciowa

FreeBSD jest jednym z najszerzej rozpowszechnionych systemów operacyjnych wykorzystywanych w wysoko wydajnych serwerach sieciowych. Rozdziały w tej części omawiają:

- Komunikację szeregową
- PPP oraz PPP over Ethernet
- Poczta elektroniczną
- Prowadzenie serwerów sieciowych
- Zapory ogniowe
- Inne zaawansowane zagadnienia związane z pracą w sieci

Poniższe rozdziały zostały tak napisane, by móc sięgnąć po nie gdy potrzebujemy danych informacji. Nie ma przymusu czytania ich w żadnej określonej kolejności, ani też przeczytania wszystkich przed rozpoczęciem pracy z FreeBSD w środowisku sieciowym.

Rozdział 22. Serial Communications

22.1. Synopsis

UNIX has always had support for serial communications. In fact, the very first UNIX machines relied on serial lines for user input and output. Things have changed a lot from the days when the average „terminal” consisted of a 10-character-per-second serial printer and a keyboard. This chapter will cover some of the ways in which FreeBSD uses serial communications.

After reading this chapter, you will know:

- How to connect terminals to your FreeBSD system.
- How to use a modem to dial out to remote hosts.
- How to allow remote users to login to your system with a modem.
- How to boot your system from a serial console.

Before reading this chapter, you should:

- Know how to configure and install a new kernel (Rozdział 8).
- Understand UNIX permissions and processes (Rozdział 3).
- Have access to the technical manual for the serial hardware (modem or multi-port card) that you would like to use with FreeBSD.

22.2. Introduction

22.2.1. Terminology

bps

Bits per Second — the rate at which data is transmitted

DTE

Data Terminal Equipment — for example, your computer

DCE

Data Communications Equipment — your modem

RS-232

EIA standard for hardware serial communications

When talking about communications data rates, this section does not use the term „baud”. Baud refers to the number of electrical state transitions that may be made in a period of time, while „bps” (bits per second) is the *correct* term to use (at least it does not seem to bother the curmudgeons quite as much).

22.2.2. Cables and Ports

To connect a modem or terminal to your FreeBSD system, you will need a serial port on your computer and the proper cable to connect to your serial device. If you are already familiar with your hardware and the cable it requires, you can safely skip this section.

22.2.2.1. Cables

There are several different kinds of serial cables. The two most common types for our purposes are null-modem cables and standard („straight”) RS-232 cables. The documentation for your hardware should describe the type of cable required.

22.2.2.1.1. Null-modem Cables

A null-modem cable passes some signals, such as „Signal Ground”, straight through, but switches other signals. For example, the „Transmitted Data” pin on one end goes to the „Received Data” pin on the other end.

You can also construct your own null-modem cable for use with terminals (e.g., for quality purposes). This table shows the RS-232C signals and the pin numbers on a DB-25 connector. Note that the standard also calls for a straight-through pin 1 to pin 1 *Protective Ground* line, but it is often omitted. Some terminals work OK using only pins 2, 3 and 7, while others require different configurations than the examples shown below.

Tabela 22-1. DB-25 to DB-25 Null-Modem Cable

Signal	Pin #		Pin #	Signal
SG	7	connects to	7	SG
TD	2	connects to	3	RD
RD	3	connects to	2	TD
RTS	4	connects to	5	CTS
CTS	5	connects to	4	RTS
DTR	20	connects to	6	DSR
DTR	20	connects to	8	DCD
DSR	6	connects to	20	DTR
DCD	8	connects to	20	DTR

Here are two other schemes more common nowadays.

Tabela 22-2. DB-9 to DB-9 Null-Modem Cable

Signal	Pin #		Pin #	Signal
RD	2	connects to	3	TD
TD	3	connects to	2	RD
DTR	4	connects to	6	DSR
DTR	4	connects to	1	DCD
SG	5	connects to	5	SG
DSR	6	connects to	4	DTR
DCD	1	connects to	4	DTR

Signal	Pin #		Pin #	Signal
RTS	7	connects to	8	CTS
CTS	8	connects to	7	RTS

Tabela 22-3. DB-9 to DB-25 Null-Modem Cable

Signal	Pin #		Pin #	Signal
RD	2	connects to	2	TD
TD	3	connects to	3	RD
DTR	4	connects to	6	DSR
DTR	4	connects to	8	DCD
SG	5	connects to	7	SG
DSR	6	connects to	20	DTR
DCD	1	connects to	20	DTR
RTS	7	connects to	5	CTS
CTS	8	connects to	4	RTS

Notatka: When one pin at one end connects to a pair of pins at the other end, it is usually implemented with one short wire between the pair of pins in their connector and a long wire to the other single pin.

The above designs seems to be the most popular. In another variation (explained in the book *RS-232 Made Easy*) SG connects to SG, TD connects to RD, RTS and CTS connect to DCD, DTR connects to DSR, and vice-versa.

22.2.2.1.2. Standard RS-232C Cables

A standard serial cable passes all of the RS-232C signals straight through. That is, the „Transmitted Data” pin on one end of the cable goes to the „Transmitted Data” pin on the other end. This is the type of cable to use to connect a modem to your FreeBSD system, and is also appropriate for some terminals.

22.2.2.2. Ports

Serial ports are the devices through which data is transferred between the FreeBSD host computer and the terminal. This section describes the kinds of ports that exist and how they are addressed in FreeBSD.

22.2.2.2.1. Kinds of Ports

Several kinds of serial ports exist. Before you purchase or construct a cable, you need to make sure it will fit the ports on your terminal and on the FreeBSD system.

Most terminals will have DB-25 ports. Personal computers, including PCs running FreeBSD, will have DB-25 or DB-9 ports. If you have a multiport serial card for your PC, you may have RJ-12 or RJ-45 ports.

See the documentation that accompanied the hardware for specifications on the kind of port in use. A visual inspection of the port often works too.

22.2.2.2. Port Names

In FreeBSD, you access each serial port through an entry in the `/dev` directory. There are two different kinds of entries:

- Call-in ports are named `/dev/ttyd N` where N is the port number, starting from zero. Generally, you use the call-in port for terminals. Call-in ports require that the serial line assert the data carrier detect (DCD) signal to work correctly.
- Call-out ports are named `/dev/cuad N` . You usually do not use the call-out port for terminals, just for modems. You may use the call-out port if the serial cable or the terminal does not support the carrier detect signal.

Notatka: Call-out ports are named `/dev/cuaa N` in FreeBSD 5.X and older.

If you have connected a terminal to the first serial port (COM1 in MS-DOS), then you will use `/dev/ttyd0` to refer to the terminal. If the terminal is on the second serial port (also known as COM2), use `/dev/ttyd1`, and so forth.

22.2.3. Kernel Configuration

FreeBSD supports four serial ports by default. In the MS-DOS world, these are known as COM1, COM2, COM3, and COM4. FreeBSD currently supports „dumb” multiport serial interface cards, such as the BocaBoard 1008 and 2016, as well as more intelligent multi-port cards such as those made by Digiboard and Stallion Technologies. However, the default kernel only looks for the standard COM ports.

To see if your kernel recognizes any of your serial ports, watch for messages while the kernel is booting, or use the `/sbin/dmesg` command to replay the kernel’s boot messages. In particular, look for messages that start with the characters `sio`.

Podpowiedź: To view just the messages that have the word `sio`, use the command:

```
# /sbin/dmesg | grep 'sio'
```

For example, on a system with four serial ports, these are the serial-port specific kernel boot messages:

```
sio0 at 0x3f8-0x3ff irq 4 on isa
sio0: type 16550A
sio1 at 0x2f8-0x2ff irq 3 on isa
sio1: type 16550A
sio2 at 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
sio3 at 0x2e8-0x2ef irq 9 on isa
sio3: type 16550A
```

If your kernel does not recognize all of your serial ports, you will probably need to configure your kernel in the `/boot/device.hints` file. You can also comment-out or completely remove lines for devices you do not have.

Please refer to the `sio(4)` manual page for more information on serial ports and multiport boards configuration. Be careful if you are using a configuration file that was previously used for a different version of FreeBSD because the device flags and the syntax have changed between versions.

Notatka: `port IO_COM1` is a substitution for `port 0x3f8`, `IO_COM2` is `0x2f8`, `IO_COM3` is `0x3e8`, and `IO_COM4` is `0x2e8`, which are fairly common port addresses for their respective serial ports; interrupts 4, 3, 5, and 9 are fairly common interrupt request lines. Also note that regular serial ports *cannot* share interrupts on ISA-bus PCs (multiport boards have on-board electronics that allow all the 16550A's on the board to share one or two interrupt request lines).

22.2.4. Device Special Files

Most devices in the kernel are accessed through „device special files”, which are located in the `/dev` directory. The `sio` devices are accessed through the `/dev/ttydN` (dial-in) and `/dev/cuadN` (call-out) devices. FreeBSD also provides initialization devices (`/dev/ttydN.init` and `/dev/cuadN.init` on FreeBSD 6.X, `/dev/ttyidN` and `/dev/cuaiaN` on FreeBSD 5.X) and locking devices (`/dev/ttydN.lock` and `/dev/cuadN.lock` on FreeBSD 6.X, `/dev/ttyldN` and `/dev/cualaN` on FreeBSD 5.X). The initialization devices are used to initialize communications port parameters each time a port is opened, such as `crtsets` for modems which use RTS/CTS signaling for flow control. The locking devices are used to lock flags on ports to prevent users or programs changing certain parameters; see the manual pages `termios(4)`, `sio(4)`, and `stty(1)` for information on the terminal settings, locking and initializing devices, and setting terminal options, respectively.

22.2.5. Serial Port Configuration

The `tydN` (or `cuadN`) device is the regular device you will want to open for your applications. When a process opens the device, it will have a default set of terminal I/O settings. You can see these settings with the command

```
# stty -a -f /dev/ttyd1
```

When you change the settings to this device, the settings are in effect until the device is closed. When it is reopened, it goes back to the default set. To make changes to the default set, you can open and adjust the settings of the „initial state” device. For example, to turn on `CLOCAL` mode, 8 bit communication, and `XON/XOFF` flow control by default for `tyd5`, type:

```
# stty -f /dev/ttyd5.init clocal cs8 ixon ixoff
```

System-wide initialization of the serial devices is controlled in `/etc/rc.d/serial`. This file affects the default settings of serial devices.

To prevent certain settings from being changed by an application, make adjustments to the „lock state” device. For example, to lock the speed of `tyd5` to 57600 bps, type:

```
# stty -f /dev/ttyd5.lock 57600
```

Now, an application that opens `tyd5` and tries to change the speed of the port will be stuck with 57600 bps.

Naturally, you should make the initial state and lock state devices writable only by the `root` account.

22.3. Terminals

Contributed by Sean Kelly.

Terminals provide a convenient and low-cost way to access your FreeBSD system when you are not at the computer's console or on a connected network. This section describes how to use terminals with FreeBSD.

22.3.1. Uses and Types of Terminals

The original UNIX systems did not have consoles. Instead, people logged in and ran programs through terminals that were connected to the computer's serial ports. It is quite similar to using a modem and terminal software to dial into a remote system to do text-only work.

Today's PCs have consoles capable of high quality graphics, but the ability to establish a login session on a serial port still exists in nearly every UNIX style operating system today; FreeBSD is no exception. By using a terminal attached to an unused serial port, you can log in and run any text program that you would normally run on the console or in an `xterm` window in the X Window System.

For the business user, you can attach many terminals to a FreeBSD system and place them on your employees' desktops. For a home user, a spare computer such as an older IBM PC or a Macintosh can be a terminal wired into a more powerful computer running FreeBSD. You can turn what might otherwise be a single-user computer into a powerful multiple user system.

For FreeBSD, there are three kinds of terminals:

- Dumb terminals
- PCs acting as terminals
- X terminals

The remaining subsections describe each kind.

22.3.1.1. Dumb Terminals

Dumb terminals are specialized pieces of hardware that let you connect to computers over serial lines. They are called „dumb” because they have only enough computational power to display, send, and receive text. You cannot run any programs on them. It is the computer to which you connect them that has all the power to run text editors, compilers, email, games, and so forth.

There are hundreds of kinds of dumb terminals made by many manufacturers, including Digital Equipment Corporation's VT-100 and Wyse's WY-75. Just about any kind will work with FreeBSD. Some high-end terminals can even display graphics, but only certain software packages can take advantage of these advanced features.

Dumb terminals are popular in work environments where workers do not need access to graphical applications such as those provided by the X Window System.

22.3.1.2. PCs Acting as Terminals

If a dumb terminal has just enough ability to display, send, and receive text, then certainly any spare personal computer can be a dumb terminal. All you need is the proper cable and some *terminal emulation* software to run on the computer.

Such a configuration is popular in homes. For example, if your spouse is busy working on your FreeBSD system's console, you can do some text-only work at the same time from a less powerful personal computer hooked up as a terminal to the FreeBSD system.

There are at least two utilities in the base-system of FreeBSD that can be used to work through a serial connection: `cu(1)` and `tip(1)`.

To connect from a client system that runs FreeBSD to the serial connection of another system, you can use:

```
# cu -l serial-port-device
```

Where „serial-port-device” is the name of a special device file denoting a serial port of your system. These device files are called `/dev/cuaaN` for FreeBSD versions older than 6.0, and `/dev/cuadN` for 6.0 and later versions.

The „N”-part of a device name is the serial port number.

Notatka: Note that device numbers in FreeBSD start from zero and not one (like they do, for instance in MS-DOS-derived systems). This means that what MS-DOS-based systems call „COM1” is usually `/dev/cuad0` in FreeBSD.

Notatka: Some people prefer to use other programs, available through the Ports Collection. The Ports include quite a few utilities which can work in ways similar to `cu(1)` and `tip(1)`, i.e. `comms/minicom`.

22.3.1.3. X Terminals

X terminals are the most sophisticated kind of terminal available. Instead of connecting to a serial port, they usually connect to a network like Ethernet. Instead of being relegated to text-only applications, they can display any X application.

We introduce X terminals just for the sake of completeness. However, this chapter does *not* cover setup, configuration, or use of X terminals.

22.3.2. Configuration

This section describes what you need to configure on your FreeBSD system to enable a login session on a terminal. It assumes you have already configured your kernel to support the serial port to which the terminal is connected—and that you have connected it.

Recall from Rozdział 12 that the `init` process is responsible for all process control and initialization at system startup. One of the tasks performed by `init` is to read the `/etc/ttys` file and start a `getty` process on the available terminals. The `getty` process is responsible for reading a login name and starting the `login` program.

Thus, to configure terminals for your FreeBSD system the following steps should be taken as `root`:

1. Add a line to `/etc/ttys` for the entry in the `/dev` directory for the serial port if it is not already there.
2. Specify that `/usr/libexec/getty` be run on the port, and specify the appropriate `getty` type from the `/etc/gettytab` file.

3. Specify the default terminal type.
4. Set the port to „on.”
5. Specify whether the port should be „secure.”
6. Force `init` to reread the `/etc/ttys` file.

As an optional step, you may wish to create a custom `getty` type for use in step 2 by making an entry in `/etc/gettytab`. This chapter does not explain how to do so; you are encouraged to see the `gettytab(5)` and the `getty(8)` manual pages for more information.

22.3.2.1. Adding an Entry to `/etc/ttys`

The `/etc/ttys` file lists all of the ports on your FreeBSD system where you want to allow logins. For example, the first virtual console `ttyv0` has an entry in this file. You can log in on the console using this entry. This file also contains entries for the other virtual consoles, serial ports, and pseudo-ttys. For a hardwired terminal, just list the serial port's `/dev` entry without the `/dev` part (for example, `/dev/ttyv0` would be listed as `ttyv0`).

A default FreeBSD install includes an `/etc/ttys` file with support for the first four serial ports: `ttyd0` through `ttyd3`. If you are attaching a terminal to one of those ports, you do not need to add another entry.

Przykład 22-1. Adding Terminal Entries to `/etc/ttys`

Suppose we would like to connect two terminals to the system: a Wyse-50 and an old 286 IBM PC running **Procomm** terminal software emulating a VT-100 terminal. We connect the Wyse to the second serial port and the 286 to the sixth serial port (a port on a multiport serial card). The corresponding entries in the `/etc/ttys` file would look like this:

```
ttyd1❶  "/usr/libexec/getty std.38400"❷  wy50❸  on❹  insecure❺
ttyd5   "/usr/libexec/getty std.19200"  vt100  on   insecure
```

- ❶ The first field normally specifies the name of the terminal special file as it is found in `/dev`.
- ❷ The second field is the command to execute for this line, which is usually `getty(8)`. `getty` initializes and opens the line, sets the speed, prompts for a user name and then executes the `login(1)` program.

The `getty` program accepts one (optional) parameter on its command line, the `getty` type. A `getty` type configures characteristics on the terminal line, like bps rate and parity. The `getty` program reads these characteristics from the file `/etc/gettytab`.

The file `/etc/gettytab` contains lots of entries for terminal lines both old and new. In almost all cases, the entries that start with the text `std` will work for hardwired terminals. These entries ignore parity. There is a `std` entry for each bps rate from 110 to 115200. Of course, you can add your own entries to this file. The `gettytab(5)` manual page provides more information.

When setting the `getty` type in the `/etc/ttys` file, make sure that the communications settings on the terminal match.

For our example, the Wyse-50 uses no parity and connects at 38400 bps. The 286 PC uses no parity and connects at 19200 bps.

- ❸ The third field is the type of terminal usually connected to that tty line. For dial-up ports, `unknown` or `dialup` is typically used in this field since users may dial up with practically any type of terminal or software. For

hardwired terminals, the terminal type does not change, so you can put a real terminal type from the `termcap(5)` database file in this field.

For our example, the Wyse-50 uses the real terminal type while the 286 PC running **Procomm** will be set to emulate at VT-100.

- ④ The fourth field specifies if the port should be enabled. Putting `on` here will have the `init` process start the program in the second field, `getty`. If you put `off` in this field, there will be no `getty`, and hence no logins on the port.
- ⑤ The final field is used to specify whether the port is secure. Marking a port as secure means that you trust it enough to allow the `root` account (or any account with a user ID of 0) to login from that port. Insecure ports do not allow `root` logins. On an insecure port, users must login from unprivileged accounts and then use `su(1)` or a similar mechanism to gain superuser privileges.

It is highly recommended that you use „insecure” even for terminals that are behind locked doors. It is quite easy to login and use `su` if you need superuser privileges.

22.3.2.2. Force `init` to Reread `/etc/ttys`

After making the necessary changes to the `/etc/ttys` file you should send a `SIGHUP` (hangup) signal to the `init` process to force it to re-read its configuration file. For example:

```
# kill -HUP 1
```

Notatka: `init` is always the first process run on a system, therefore it will always have PID 1.

If everything is set up correctly, all cables are in place, and the terminals are powered up, then a `getty` process should be running on each terminal and you should see login prompts on your terminals at this point.

22.3.3. Troubleshooting Your Connection

Even with the most meticulous attention to detail, something could still go wrong while setting up a terminal. Here is a list of symptoms and some suggested fixes.

22.3.3.1. No Login Prompt Appears

Make sure the terminal is plugged in and powered up. If it is a personal computer acting as a terminal, make sure it is running terminal emulation software on the correct serial port.

Make sure the cable is connected firmly to both the terminal and the FreeBSD computer. Make sure it is the right kind of cable.

Make sure the terminal and FreeBSD agree on the bps rate and parity settings. If you have a video display terminal, make sure the contrast and brightness controls are turned up. If it is a printing terminal, make sure paper and ink are in good supply.

Make sure that a `getty` process is running and serving the terminal. For example, to get a list of running `getty` processes with `ps`, type:

```
# ps -axww|grep getty
```

You should see an entry for the terminal. For example, the following display shows that a `getty` is running on the second serial port `ttyd1` and is using the `std.38400` entry in `/etc/gettytab`:

```
22189  d1  Is+      0:00.03 /usr/libexec/getty std.38400 ttyd1
```

If no `getty` process is running, make sure you have enabled the port in `/etc/ttys`. Also remember to run `kill -HUP 1` after modifying the `ttys` file.

If the `getty` process is running but the terminal still does not display a login prompt, or if it displays a prompt but will not allow you to type, your terminal or cable may not support hardware handshaking. Try changing the entry in `/etc/ttys` from `std.38400` to `3wire.38400` remember to run `kill -HUP 1` after modifying `/etc/ttys`). The `3wire` entry is similar to `std`, but ignores hardware handshaking. You may need to reduce the baud rate or enable software flow control when using `3wire` to prevent buffer overflows.

22.3.3.2. If Garbage Appears Instead of a Login Prompt

Make sure the terminal and FreeBSD agree on the bps rate and parity settings. Check the `getty` processes to make sure the correct `getty` type is in use. If not, edit `/etc/ttys` and run `kill -HUP 1`.

22.3.3.3. Characters Appear Doubled; the Password Appears When Typed

Switch the terminal (or the terminal emulation software) from „half duplex” or „local echo” to „full duplex.”

22.4. Dial-in Service

Contributed by Guy Helmer. Additions by Sean Kelly.

Configuring your FreeBSD system for dial-in service is very similar to connecting terminals except that you are dealing with modems instead of terminals.

22.4.1. External vs. Internal Modems

External modems seem to be more convenient for dial-up, because external modems often can be semi-permanently configured via parameters stored in non-volatile RAM and they usually provide lighted indicators that display the state of important RS-232 signals. Blinking lights impress visitors, but lights are also very useful to see whether a modem is operating properly.

Internal modems usually lack non-volatile RAM, so their configuration may be limited only to setting DIP switches. If your internal modem has any signal indicator lights, it is probably difficult to view the lights when the system's cover is in place.

22.4.1.1. Modems and Cables

If you are using an external modem, then you will of course need the proper cable. A standard RS-232C serial cable should suffice as long as all of the normal signals are wired:

Tabela 22-4. Signal Names

Acronyms	Names
RD	Received Data
TD	Transmitted Data
DTR	Data Terminal Ready
DSR	Data Set Ready
DCD	Data Carrier Detect (RS-232's Received Line Signal Detector)
SG	Signal Ground
RTS	Request to Send
CTS	Clear to Send

FreeBSD needs the RTS and CTS signals for flow control at speeds above 2400 bps, the CD signal to detect when a call has been answered or the line has been hung up, and the DTR signal to reset the modem after a session is complete. Some cables are wired without all of the needed signals, so if you have problems, such as a login session not going away when the line hangs up, you may have a problem with your cable.

Like other UNIX like operating systems, FreeBSD uses the hardware signals to find out when a call has been answered or a line has been hung up and to hangup and reset the modem after a call. FreeBSD avoids sending commands to the modem or watching for status reports from the modem. If you are familiar with connecting modems to PC-based bulletin board systems, this may seem awkward.

22.4.2. Serial Interface Considerations

FreeBSD supports NS8250-, NS16450-, NS16550-, and NS16550A-based EIA RS-232C (CCITT V.24) communications interfaces. The 8250 and 16450 devices have single-character buffers. The 16550 device provides a 16-character buffer, which allows for better system performance. (Bugs in plain 16550's prevent the use of the 16-character buffer, so use 16550A's if possible). Because single-character-buffer devices require more work by the operating system than the 16-character-buffer devices, 16550A-based serial interface cards are much preferred. If the system has many active serial ports or will have a heavy load, 16550A-based cards are better for low-error-rate communications.

22.4.3. Quick Overview

As with terminals, `init` spawns a `getty` process for each configured serial port for dial-in connections. For example, if a modem is attached to `/dev/ttyd0`, the command `ps ax` might show this:

```
4850 ?? I      0:00.09 /usr/libexec/getty V19200 ttyd0
```

When a user dials the modem's line and the modems connect, the CD (Carrier Detect) line is reported by the modem. The kernel notices that carrier has been detected and completes `getty`'s open of the port. `getty` sends a `login:` prompt at the specified initial line speed. `getty` watches to see if legitimate characters are received, and, in a typical configuration, if it finds junk (probably due to the modem's connection speed being different than `getty`'s speed), `getty` tries adjusting the line speeds until it receives reasonable characters.

After the user enters his/her login name, `getty` executes `/usr/bin/login`, which completes the login by asking for the user's password and then starting the user's shell.

22.4.4. Configuration Files

There are three system configuration files in the `/etc` directory that you will probably need to edit to allow dial-up access to your FreeBSD system. The first, `/etc/gettytab`, contains configuration information for the `/usr/libexec/getty` daemon. Second, `/etc/ttys` holds information that tells `/sbin/init` what `tty` devices should have `getty` processes running on them. Lastly, you can place port initialization commands in the `/etc/rc.d/serial` script.

There are two schools of thought regarding dial-up modems on UNIX. One group likes to configure their modems and systems so that no matter at what speed a remote user dials in, the local computer-to-modem RS-232 interface runs at a locked speed. The benefit of this configuration is that the remote user always sees a system login prompt immediately. The downside is that the system does not know what a user's true data rate is, so full-screen programs like Emacs will not adjust their screen-painting methods to make their response better for slower connections.

The other school configures their modems' RS-232 interface to vary its speed based on the remote user's connection speed. For example, V.32bis (14.4 Kbps) connections to the modem might make the modem run its RS-232 interface at 19.2 Kbps, while 2400 bps connections make the modem's RS-232 interface run at 2400 bps. Because `getty` does not understand any particular modem's connection speed reporting, `getty` gives a `login:` message at an initial speed and watches the characters that come back in response. If the user sees junk, it is assumed that they know they should press the Enter key until they see a recognizable prompt. If the data rates do not match, `getty` sees anything the user types as „junk”, tries going to the next speed and gives the `login:` prompt again. This procedure can continue ad nauseam, but normally only takes a keystroke or two before the user sees a good prompt. Obviously, this login sequence does not look as clean as the former „locked-speed” method, but a user on a low-speed connection should receive better interactive response from full-screen programs.

This section will try to give balanced configuration information, but is biased towards having the modem's data rate follow the connection rate.

22.4.4.1. `/etc/gettytab`

`/etc/gettytab` is a `termcap(5)`-style file of configuration information for `getty(8)`. Please see the `gettytab(5)` manual page for complete information on the format of the file and the list of capabilities.

22.4.4.1.1. *Locked-speed Config*

If you are locking your modem's data communications rate at a particular speed, you probably will not need to make any changes to `/etc/gettytab`.

22.4.4.1.2. Matching-speed Config

You will need to set up an entry in `/etc/gettytab` to give `getty` information about the speeds you wish to use for your modem. If you have a 2400 bps modem, you can probably use the existing `D2400` entry.

```
#
# Fast dialup terminals, 2400/1200/300 rotary (can start either way)
#
D2400|d2400|Fast-Dial-2400:\
        :nx=D1200:tc=2400-baud:
3|D1200|Fast-Dial-1200:\
        :nx=D300:tc=1200-baud:
5|D300|Fast-Dial-300:\
        :nx=D2400:tc=300-baud:
```

If you have a higher speed modem, you will probably need to add an entry in `/etc/gettytab`; here is an entry you could use for a 14.4 Kbps modem with a top interface speed of 19.2 Kbps:

```
#
# Additions for a V.32bis Modem
#
um|V300|High Speed Modem at 300,8-bit:\
        :nx=V19200:tc=std.300:
un|V1200|High Speed Modem at 1200,8-bit:\
        :nx=V300:tc=std.1200:
uo|V2400|High Speed Modem at 2400,8-bit:\
        :nx=V1200:tc=std.2400:
up|V9600|High Speed Modem at 9600,8-bit:\
        :nx=V2400:tc=std.9600:
uq|V19200|High Speed Modem at 19200,8-bit:\
        :nx=V9600:tc=std.19200:
```

This will result in 8-bit, no parity connections.

The example above starts the communications rate at 19.2 Kbps (for a V.32bis connection), then cycles through 9600 bps (for V.32), 2400 bps, 1200 bps, 300 bps, and back to 19.2 Kbps. Communications rate cycling is implemented with the `nx=` („next table”) capability. Each of the lines uses a `tc=` („table continuation”) entry to pick up the rest of the „standard” settings for a particular data rate.

If you have a 28.8 Kbps modem and/or you want to take advantage of compression on a 14.4 Kbps modem, you need to use a higher communications rate than 19.2 Kbps. Here is an example of a `gettytab` entry starting a 57.6 Kbps:

```
#
# Additions for a V.32bis or V.34 Modem
# Starting at 57.6 Kbps
#
vm|VH300|Very High Speed Modem at 300,8-bit:\
        :nx=VH57600:tc=std.300:
vn|VH1200|Very High Speed Modem at 1200,8-bit:\
        :nx=VH300:tc=std.1200:
vo|VH2400|Very High Speed Modem at 2400,8-bit:\
        :nx=VH1200:tc=std.2400:
vp|VH9600|Very High Speed Modem at 9600,8-bit:\
        :nx=VH2400:tc=std.9600:
```

```
vq|VH57600|Very High Speed Modem at 57600,8-bit:\
      :nx=VH9600:tc=std.57600:
```

If you have a slow CPU or a heavily loaded system and do not have 16550A-based serial ports, you may receive `sio „silo”` errors at 57.6 Kbps.

22.4.4.2. `/etc/ttys`

Configuration of the `/etc/ttys` file was covered in [Przykład 22-1](#). Configuration for modems is similar but we must pass a different argument to `getty` and specify a different terminal type. The general format for both locked-speed and matching-speed configurations is:

```
ttyd0    "/usr/libexec/getty xxx"    dialup on
```

The first item in the above line is the device special file for this entry — `ttyd0` means `/dev/ttyd0` is the file that this `getty` will be watching. The second item, `"/usr/libexec/getty xxx"` (`xxx` will be replaced by the initial `gettytab` capability) is the process `init` will run on the device. The third item, `dialup`, is the default terminal type. The fourth parameter, `on`, indicates to `init` that the line is operational. There can be a fifth parameter, `secure`, but it should only be used for terminals which are physically secure (such as the system console).

The default terminal type (`dialup` in the example above) may depend on local preferences. `dialup` is the traditional default terminal type on dial-up lines so that users may customize their login scripts to notice when the terminal is `dialup` and automatically adjust their terminal type. However, the author finds it easier at his site to specify `vt102` as the default terminal type, since the users just use VT102 emulation on their remote systems.

After you have made changes to `/etc/ttys`, you may send the `init` process a HUP signal to re-read the file. You can use the command

```
# kill -HUP 1
```

to send the signal. If this is your first time setting up the system, you may want to wait until your modem(s) are properly configured and connected before signaling `init`.

22.4.4.2.1. Locked-speed Config

For a locked-speed configuration, your `ttys` entry needs to have a fixed-speed entry provided to `getty`. For a modem whose port speed is locked at 19.2 Kbps, the `ttys` entry might look like this:

```
ttyd0    "/usr/libexec/getty std.19200"    dialup on
```

If your modem is locked at a different data rate, substitute the appropriate value for `std.speed` instead of `std.19200`. Make sure that you use a valid type listed in `/etc/gettytab`.

22.4.4.2.2. Matching-speed Config

In a matching-speed configuration, your `ttys` entry needs to reference the appropriate beginning „auto-baud” (sic) entry in `/etc/gettytab`. For example, if you added the above suggested entry for a matching-speed modem that starts at 19.2 Kbps (the `gettytab` entry containing the `V19200` starting point), your `ttys` entry might look like this:

```
ttyd0    "/usr/libexec/getty V19200"    dialup on
```

22.4.4.3. /etc/rc.d/serial

High-speed modems, like V.32, V.32bis, and V.34 modems, need to use hardware (RTS/CTS) flow control. You can add `stty` commands to `/etc/rc.d/serial` to set the hardware flow control flag in the FreeBSD kernel for the modem ports.

For example to set the `termios` flag `crtsets` on serial port #1's (COM2) dial-in and dial-out initialization devices, the following lines could be added to `/etc/rc.d/serial`:

```
# Serial port initial configuration
stty -f /dev/ttydl.init crtsets
stty -f /dev/cuadl.init crtsets
```

22.4.5. Modem Settings

If you have a modem whose parameters may be permanently set in non-volatile RAM, you will need to use a terminal program (such as `Telx` under MS-DOS or `tip` under FreeBSD) to set the parameters. Connect to the modem using the same communications speed as the initial speed `getty` will use and configure the modem's non-volatile RAM to match these requirements:

- CD asserted when connected
- DTR asserted for operation; dropping DTR hangs up line and resets modem
- CTS transmitted data flow control
- Disable XON/XOFF flow control
- RTS received data flow control
- Quiet mode (no result codes)
- No command echo

Please read the documentation for your modem to find out what commands and/or DIP switch settings you need to give it.

For example, to set the above parameters on a U.S. Robotics® Sportster® 14,400 external modem, one could give these commands to the modem:

```
ATZ
AT&C1&D2&H1&I0&R2&W
```

You might also want to take this opportunity to adjust other settings in the modem, such as whether it will use V.42bis and/or MNP5 compression.

The U.S. Robotics Sportster 14,400 external modem also has some DIP switches that need to be set; for other modems, perhaps you can use these settings as an example:

- Switch 1: UP — DTR Normal
- Switch 2: N/A (Verbal Result Codes/Numeric Result Codes)
- Switch 3: UP — Suppress Result Codes

- Switch 4: DOWN — No echo, offline commands
- Switch 5: UP — Auto Answer
- Switch 6: UP — Carrier Detect Normal
- Switch 7: UP — Load NVRAM Defaults
- Switch 8: N/A (Smart Mode/Dumb Mode)

Result codes should be disabled/suppressed for dial-up modems to avoid problems that can occur if `getty` mistakenly gives a `login:` prompt to a modem that is in command mode and the modem echoes the command or returns a result code. This sequence can result in an extended, silly conversation between `getty` and the modem.

22.4.5.1. Locked-speed Config

For a locked-speed configuration, you will need to configure the modem to maintain a constant modem-to-computer data rate independent of the communications rate. On a U.S. Robotics Sportster 14,400 external modem, these commands will lock the modem-to-computer data rate at the speed used to issue the commands:

```
ATZ
AT&B1&W
```

22.4.5.2. Matching-speed Config

For a variable-speed configuration, you will need to configure your modem to adjust its serial port data rate to match the incoming call rate. On a U.S. Robotics Sportster 14,400 external modem, these commands will lock the modem's error-corrected data rate to the speed used to issue the commands, but allow the serial port rate to vary for non-error-corrected connections:

```
ATZ
AT&B2&W
```

22.4.5.3. Checking the Modem's Configuration

Most high-speed modems provide commands to view the modem's current operating parameters in a somewhat human-readable fashion. On the U.S. Robotics Sportster 14,400 external modems, the command `ATI5` displays the settings that are stored in the non-volatile RAM. To see the true operating parameters of the modem (as influenced by the modem's DIP switch settings), use the commands `ATZ` and then `ATI4`.

If you have a different brand of modem, check your modem's manual to see how to double-check your modem's configuration parameters.

22.4.6. Troubleshooting

Here are a few steps you can follow to check out the dial-up modem on your system.

22.4.6.1. Checking Out the FreeBSD System

Hook up your modem to your FreeBSD system, boot the system, and, if your modem has status indication lights, watch to see whether the modem's DTR indicator lights when the `login:` prompt appears on the system's console — if it lights up, that should mean that FreeBSD has started a `getty` process on the appropriate communications port and is waiting for the modem to accept a call.

If the DTR indicator does not light, login to the FreeBSD system through the console and issue a `ps ax` to see if FreeBSD is trying to run a `getty` process on the correct port. You should see lines like these among the processes displayed:

```
114 ?? I      0:00.10 /usr/libexec/getty V19200 ttyd0
115 ?? I      0:00.10 /usr/libexec/getty V19200 ttyd1
```

If you see something different, like this:

```
114 d0 I      0:00.10 /usr/libexec/getty V19200 ttyd0
```

and the modem has not accepted a call yet, this means that `getty` has completed its open on the communications port. This could indicate a problem with the cabling or a mis-configured modem, because `getty` should not be able to open the communications port until CD (carrier detect) has been asserted by the modem.

If you do not see any `getty` processes waiting to open the desired `ttyn` port, double-check your entries in `/etc/ttys` to see if there are any mistakes there. Also, check the log file `/var/log/messages` to see if there are any log messages from `init` or `getty` regarding any problems. If there are any messages, triple-check the configuration files `/etc/ttys` and `/etc/gettytab`, as well as the appropriate device special files `/dev/ttydN`, for any mistakes, missing entries, or missing device special files.

22.4.6.2. Try Dialing In

Try dialing into the system; be sure to use 8 bits, no parity, and 1 stop bit on the remote system. If you do not get a prompt right away, or get garbage, try pressing Enter about once per second. If you still do not see a `login:` prompt after a while, try sending a `BREAK`. If you are using a high-speed modem to do the dialing, try dialing again after locking the dialing modem's interface speed (via `AT&B1` on a U.S. Robotics Sportster modem, for example).

If you still cannot get a `login:` prompt, check `/etc/gettytab` again and double-check that

- The initial capability name specified in `/etc/ttys` for the line matches a name of a capability in `/etc/gettytab`
- Each `nx=` entry matches another `gettytab` capability name
- Each `tc=` entry matches another `gettytab` capability name

If you dial but the modem on the FreeBSD system will not answer, make sure that the modem is configured to answer the phone when DTR is asserted. If the modem seems to be configured correctly, verify that the DTR line is asserted by checking the modem's indicator lights (if it has any).

If you have gone over everything several times and it still does not work, take a break and come back to it later. If it still does not work, perhaps you can send an electronic mail message to the Ogólna lista dyskusyjna FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>) describing your modem and your problem, and the good folks on the list will try to help.

22.5. Dial-out Service

The following are tips for getting your host to be able to connect over the modem to another computer. This is appropriate for establishing a terminal session with a remote host.

This is useful to log onto a BBS.

This kind of connection can be extremely helpful to get a file on the Internet if you have problems with PPP. If you need to FTP something and PPP is broken, use the terminal session to FTP it. Then use `zmodem` to transfer it to your machine.

22.5.1. My Stock Hayes Modem Is Not Supported, What Can I Do?

Actually, the manual page for `tip` is out of date. There is a generic Hayes dialer already built in. Just use `at=hayes` in your `/etc/remote` file.

The Hayes driver is not smart enough to recognize some of the advanced features of newer modems—messages like `BUSY`, `NO DIALTONE`, or `CONNECT 115200` will just confuse it. You should turn those messages off when you use `tip` (using `ATX0&W`).

Also, the dial timeout for `tip` is 60 seconds. Your modem should use something less, or else `tip` will think there is a communication problem. Try `ATS7=45&W`.

Notatka: As shipped, `tip` does not yet support Hayes modems fully. The solution is to edit the file `tipconf.h` in the directory `/usr/src/usr.bin/tip/tip`. Obviously you need the source distribution to do this.

Edit the line `#define HAYES 0` to `#define HAYES 1`. Then make `make` and `make install`. Everything works nicely after that.

22.5.2. How Am I Expected to Enter These AT Commands?

Make what is called a „direct” entry in your `/etc/remote` file. For example, if your modem is hooked up to the first serial port, `/dev/cuad0`, then put in the following line:

```
cuad0:dv=/dev/cuad0:br#19200:pa=none
```

Use the highest bps rate your modem supports in the `br` capability. Then, type `tip cuad0` and you will be connected to your modem.

Or use `cu` as `root` with the following command:

```
# cu -lline -sspeed
```

`line` is the serial port (e.g. `/dev/cuad0`) and `speed` is the speed (e.g. `57600`). When you are done entering the AT commands hit `~`, to exit.

22.5.3. The @ Sign for the pn Capability Does Not Work!

The `@` sign in the phone number capability tells `tip` to look in `/etc/phones` for a phone number. But the `@` sign is also a special character in capability files like `/etc/remote`. Escape it with a backslash:

pn=\@

22.5.4. How Can I Dial a Phone Number on the Command Line?

Put what is called a „generic” entry in your `/etc/remote` file. For example:

```
tip115200|Dial any phone number at 115200 bps:\
      :dv=/dev/cuad0:br#115200:at=hayes:pa=none:du:
tip57600|Dial any phone number at 57600 bps:\
      :dv=/dev/cuad0:br#57600:at=hayes:pa=none:du:
```

Then you can do things like:

```
# tip -115200 5551234
```

If you prefer `cu` over `tip`, use a generic `cu` entry:

```
cu115200|Use cu to dial any number at 115200bps:\
      :dv=/dev/cuad1:br#57600:at=hayes:pa=none:du:
```

and type:

```
# cu 5551234 -s 115200
```

22.5.5. Do I Have to Type in the bps Rate Every Time I Do That?

Put in an entry for `tip1200` or `cu1200`, but go ahead and use whatever bps rate is appropriate with the `br` capability. `tip` thinks a good default is 1200 bps which is why it looks for a `tip1200` entry. You do not have to use 1200 bps, though.

22.5.6. I Access a Number of Hosts Through a Terminal Server

Rather than waiting until you are connected and typing `CONNECT <host>` each time, use `tip`’s `cm` capability. For example, these entries in `/etc/remote`:

```
pain|pain.deep13.com|Forrester’s machine:\
      :cm=CONNECT pain\n:tc=deep13:
muffin|muffin.deep13.com|Frank’s machine:\
      :cm=CONNECT muffin\n:tc=deep13:
deep13:Gizmonics Institute terminal server:\
      :dv=/dev/cuad2:br#38400:at=hayes:du:pa=none:pn=5551234:
```

will let you type `tip pain` or `tip muffin` to connect to the hosts `pain` or `muffin`, and `tip deep13` to get to the terminal server.

22.5.7. Can Tip Try More Than One Line for Each Site?

This is often a problem where a university has several modem lines and several thousand students trying to use them. Make an entry for your university in `/etc/remote` and use `@` for the `pn` capability:

```
big-university:\
    :pn=\@:tc=dialout
dialout:\
    :dv=/dev/cuad3:br#9600:at=courier:du:pa=none:
```

Then, list the phone numbers for the university in `/etc/phones`:

```
big-university 5551111
big-university 5551112
big-university 5551113
big-university 5551114
```

`tip` will try each one in the listed order, then give up. If you want to keep retrying, run `tip` in a while loop.

22.5.8. Why Do I Have to Hit Ctrl+P Twice to Send Ctrl+P Once?

Ctrl+P is the default „force” character, used to tell `tip` that the next character is literal data. You can set the force character to any other character with the `~s` escape, which means „set a variable.”

Type `~sforce=single-char` followed by a newline. *single-char* is any single character. If you leave out *single-char*, then the force character is the nul character, which you can get by typing **Ctrl+2** or **Ctrl+Space**. A pretty good value for *single-char* is **Shift+Ctrl+6**, which is only used on some terminal servers.

You can have the force character be whatever you want by specifying the following in your `$HOME/.tiprc` file:

```
force=<single-char>
```

22.5.9. Suddenly Everything I Type Is in Upper Case??

You must have pressed **Ctrl+A**, `tip`’s „raise character,” specially designed for people with broken caps-lock keys. Use `~s` as above and set the variable `raisechar` to something reasonable. In fact, you can set it to the same as the force character, if you never expect to use either of these features.

Here is a sample `.tiprc` file perfect for **Emacs** users who need to type **Ctrl+2** and **Ctrl+A** a lot:

```
force=^^
raisechar=^^
```

The `^^` is **Shift+Ctrl+6**.

22.5.10. How Can I Do File Transfers with `tip`?

If you are talking to another UNIX system, you can send and receive files with `~p` (put) and `~t` (take). These commands run `cat` and `echo` on the remote system to accept and send files. The syntax is:

`~p local-file [remote-file]`

`~t remote-file [local-file]`

There is no error checking, so you probably should use another protocol, like `zmodem`.

22.5.11. How Can I Run `zmodem` with `tip`?

To receive files, start the sending program on the remote end. Then, type `~C rz` to begin receiving them locally.

To send files, start the receiving program on the remote end. Then, type `~C sz files` to send them to the remote system.

22.6. Setting Up the Serial Console

Contributed by Kazutaka YOKOTA. Based on a document by Bill Paul.

22.6.1. Introduction

FreeBSD has the ability to boot on a system with only a dumb terminal on a serial port as a console. Such a configuration should be useful for two classes of people: system administrators who wish to install FreeBSD on machines that have no keyboard or monitor attached, and developers who want to debug the kernel or device drivers.

As described in Rozdział 12, FreeBSD employs a three stage bootstrap. The first two stages are in the boot block code which is stored at the beginning of the FreeBSD slice on the boot disk. The boot block will then load and run the boot loader (`/boot/loader`) as the third stage code.

In order to set up the serial console you must configure the boot block code, the boot loader code and the kernel.

22.6.2. Serial Console Configuration, Terse Version

This section assumes that you are using the default setup and just want a fast overview of setting up the serial console.

1. Connect the serial cable to COM1 and the controlling terminal.
2. To see all boot messages on the serial console, issue the following command while logged in as the superuser:

```
# echo 'console="comconsole"' >> /boot/loader.conf
```
3. Edit `/etc/ttys` and change `off` to `on` and `dialup` to `vt100` for the `ttyd0` entry. Otherwise a password will not be required to connect via the serial console, resulting in a potential security hole.

4.

Reboot the system to see if the changes took effect.

If a different configuration is required, a more in depth configuration explanation exists in Sekcja 22.6.3.

22.6.3. Serial Console Configuration

1. Prepare a serial cable.

You will need either a null-modem cable or a standard serial cable and a null-modem adapter. See Sekcja 22.2.2 for a discussion on serial cables.

2. Unplug your keyboard.

Most PC systems probe for the keyboard during the Power-On Self-Test (POST) and will generate an error if the keyboard is not detected. Some machines complain loudly about the lack of a keyboard and will not continue to boot until it is plugged in.

If your computer complains about the error, but boots anyway, then you do not have to do anything special.

(Some machines with Phoenix BIOS installed merely say `Keyboard failed` and continue to boot normally.)

If your computer refuses to boot without a keyboard attached then you will have to configure the BIOS so that it ignores this error (if it can). Consult your motherboard's manual for details on how to do this.

Podpowiedź: Set the keyboard to „Not installed” in the BIOS setup. You will still be able to use your keyboard. All this does is tell the BIOS not to probe for a keyboard at power-on. Your BIOS should not complain if the keyboard is absent. You can leave the keyboard plugged in even with this flag set to „Not installed” and the keyboard will still work.

Notatka: If your system has a PS/2® mouse, chances are very good that you may have to unplug your mouse as well as your keyboard. This is because PS/2 mice share some hardware with the keyboard and leaving the mouse plugged in can fool the keyboard probe into thinking the keyboard is still there. It is said that a Gateway 2000 Pentium 90 MHz system with an AMI BIOS that behaves this way. In general, this is not a problem since the mouse is not much good without the keyboard anyway.

3. Plug a dumb terminal into COM1 (`si00`).

If you do not have a dumb terminal, you can use an old PC/XT with a modem program, or the serial port on another UNIX box. If you do not have a COM1 (`si00`), get one. At this time, there is no way to select a port other than COM1 for the boot blocks without recompiling the boot blocks. If you are already using COM1 for another device, you will have to temporarily remove that device and install a new boot block and kernel once you get FreeBSD up and running. (It is assumed that COM1 will be available on a file/compute/terminal server anyway; if you really need COM1 for something else (and you cannot switch that something else to COM2 (`si01`)), then you probably should not even be bothering with all this in the first place.)

4. Make sure the configuration file of your kernel has appropriate flags set for COM1 (`si00`).

Relevant flags are:

`0x10`

Enables console support for this unit. The other console flags are ignored unless this is set. Currently, at most one unit can have console support; the first one (in config file order) with this flag set is preferred. This option alone will not make the serial port the console. Set the following flag or use the `-h` option described below, together with this flag.

0x20

Forces this unit to be the console (unless there is another higher priority console), regardless of the `-h` option discussed below. The flag 0x20 must be used together with the 0x10 flag.

0x40

Reserves this unit (in conjunction with 0x10) and makes the unit unavailable for normal access. You should not set this flag to the serial port unit which you want to use as the serial console. The only use of this flag is to designate the unit for kernel remote debugging. See *The Developer's Handbook* (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/books/developers-handbook/index.html) for more information on remote debugging.

Example:

```
device sio0 at isa? port IO_COM1 flags 0x10 irq 4
```

See the `sio(4)` manual page for more details.

If the flags were not set, you need to run UserConfig (on a different console) or recompile the kernel.

5. Create `boot.config` in the root directory of the a partition on the boot drive.

This file will instruct the boot block code how you would like to boot the system. In order to activate the serial console, you need one or more of the following options—if you want multiple options, include them all on the same line:

`-h`

Toggles internal and serial consoles. You can use this to switch console devices. For instance, if you boot from the internal (video) console, you can use `-h` to direct the boot loader and the kernel to use the serial port as its console device. Alternatively, if you boot from the serial port, you can use the `-h` to tell the boot loader and the kernel to use the video display as the console instead.

`-D`

Toggles single and dual console configurations. In the single configuration the console will be either the internal console (video display) or the serial port, depending on the state of the `-h` option above. In the dual console configuration, both the video display and the serial port will become the console at the same time, regardless of the state of the `-h` option. However, note that the dual console configuration takes effect only during the boot block is running. Once the boot loader gets control, the console specified by the `-h` option becomes the only console.

`-P`

Makes the boot block probe the keyboard. If no keyboard is found, the `-D` and `-h` options are automatically set.

Notatka: Due to space constraints in the current version of the boot blocks, the `-P` option is capable of detecting extended keyboards only. Keyboards with less than 101 keys (and without F11 and F12 keys) may not be detected. Keyboards on some laptop computers may not be properly found because of this limitation. If this is the case with your system, you have to abandon using the `-P` option. Unfortunately there is no workaround for this problem.

Use either the `-P` option to select the console automatically, or the `-h` option to activate the serial console.

You may include other options described in `boot(8)` as well.

The options, except for `-P`, will be passed to the boot loader (`/boot/loader`). The boot loader will determine which of the internal video or the serial port should become the console by examining the state of the `-h` option alone. This means that if you specify the `-D` option but not the `-h` option in `/boot.config`, you can use the serial port as the console only during the boot block; the boot loader will use the internal video display as the console.

6. Boot the machine.

When you start your FreeBSD box, the boot blocks will echo the contents of `/boot.config` to the console. For example:

```
/boot.config: -P
Keyboard: no
```

The second line appears only if you put `-P` in `/boot.config` and indicates presence/absence of the keyboard. These messages go to either serial or internal console, or both, depending on the option in `/boot.config`.

Options	Message goes to
none	internal console
<code>-h</code>	serial console
<code>-D</code>	serial and internal consoles
<code>-Dh</code>	serial and internal consoles
<code>-P</code> , keyboard present	internal console
<code>-P</code> , keyboard absent	serial console

After the above messages, there will be a small pause before the boot blocks continue loading the boot loader and before any further messages printed to the console. Under normal circumstances, you do not need to interrupt the boot blocks, but you may want to do so in order to make sure things are set up correctly.

Hit any key, other than Enter, at the console to interrupt the boot process. The boot blocks will then prompt you for further action. You should now see something like:

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Verify the above message appears on either the serial or internal console or both, according to the options you put in `/boot.config`. If the message appears in the correct console, hit Enter to continue the boot process.

If you want the serial console but you do not see the prompt on the serial terminal, something is wrong with your settings. In the meantime, you enter `-h` and hit Enter/Return (if possible) to tell the boot block (and then the boot loader and the kernel) to choose the serial port for the console. Once the system is up, go back and check what went wrong.

After the boot loader is loaded and you are in the third stage of the boot process you can still switch between the internal console and the serial console by setting appropriate environment variables in the boot loader. See [Sekcja 22.6.6](#).

22.6.4. Summary

Here is the summary of various settings discussed in this section and the console eventually selected.

22.6.4.1. Case 1: You Set the Flags to 0x10 for `sio0`

```
device sio0 at isa? port IO_COM1 flags 0x10 irq 4
```

Options in <code>/boot.config</code>	Console during boot blocks	Console during boot loader	Console in kernel
nothing	internal	internal	internal
-h	serial	serial	serial
-D	serial and internal	internal	internal
-Dh	serial and internal	serial	serial
-P, keyboard present	internal	internal	internal
-P, keyboard absent	serial and internal	serial	serial

22.6.4.2. Case 2: You Set the Flags to 0x30 for `sio0`

```
device sio0 at isa? port IO_COM1 flags 0x30 irq 4
```

Options in <code>/boot.config</code>	Console during boot blocks	Console during boot loader	Console in kernel
nothing	internal	internal	serial
-h	serial	serial	serial
-D	serial and internal	internal	serial
-Dh	serial and internal	serial	serial
-P, keyboard present	internal	internal	serial
-P, keyboard absent	serial and internal	serial	serial

22.6.5. Tips for the Serial Console

22.6.5.1. Setting a Faster Serial Port Speed

By default, the serial port settings are: 9600 baud, 8 bits, no parity, and 1 stop bit. If you wish to change the speed, you need to recompile at least the boot blocks. Add the following line to `/etc/make.conf` and compile new boot blocks:

```
BOOT_COMCONSOLE_SPEED=19200
```

See Sekcja 22.6.5.2 for detailed instructions about building and installing new boot blocks.

If the serial console is configured in some other way than by booting with `-h`, or if the serial console used by the kernel is different from the one used by the boot blocks, then you must also add the following option to the kernel

configuration file and compile a new kernel:

```
options CONSPEED=19200
```

22.6.5.2. Using Serial Port Other Than `sio0` for the Console

Using a port other than `sio0` as the console requires some recompiling. If you want to use another serial port for whatever reasons, recompile the boot blocks, the boot loader and the kernel as follows.

1. Get the kernel source. (See Rozdział 21)
2. Edit `/etc/make.conf` and set `BOOT_COMCONSOLE_PORT` to the address of the port you want to use (0x3F8, 0x2F8, 0x3E8 or 0x2E8). Only `sio0` through `sio3` (COM1 through COM4) can be used; multiport serial cards will not work. No interrupt setting is needed.

3. Create a custom kernel configuration file and add appropriate flags for the serial port you want to use. For example, if you want to make `sio1` (COM2) the console:

```
device sio1 at isa? port IO_COM2 flags 0x10 irq 3
```

or

```
device sio1 at isa? port IO_COM2 flags 0x30 irq 3
```

The console flags for the other serial ports should not be set.

4. Recompile and install the boot blocks and the boot loader:

```
# cd /sys/boot
# make clean
# make
# make install
```

5. Rebuild and install the kernel.
6. Write the boot blocks to the boot disk with `bsdlable(8)` and boot from the new kernel.

22.6.5.3. Entering the DDB Debugger from the Serial Line

If you wish to drop into the kernel debugger from the serial console (useful for remote diagnostics, but also dangerous if you generate a spurious `BREAK` on the serial port!) then you should compile your kernel with the following options:

```
options BREAK_TO_DEBUGGER
options DDB
```

22.6.5.4. Getting a Login Prompt on the Serial Console

While this is not required, you may wish to get a *login* prompt over the serial line, now that you can see boot messages and can enter the kernel debugging session through the serial console. Here is how to do it.

Open the file `/etc/ttys` with an editor and locate the lines:

```
tttyd0 "/usr/libexec/getty std.9600" unknown off secure
```

```
ttyd1 "/usr/libexec/getty std.9600" unknown off secure
ttyd2 "/usr/libexec/getty std.9600" unknown off secure
ttyd3 "/usr/libexec/getty std.9600" unknown off secure
```

`ttyd0` through `ttyd3` corresponds to `COM1` through `COM4`. Change `off` to `on` for the desired port. If you have changed the speed of the serial port, you need to change `std.9600` to match the current setting, e.g. `std.19200`.

You may also want to change the terminal type from `unknown` to the actual type of your serial terminal.

After editing the file, you must `kill -HUP 1` to make this change take effect.

22.6.6. Changing Console from the Boot Loader

Previous sections described how to set up the serial console by tweaking the boot block. This section shows that you can specify the console by entering some commands and environment variables in the boot loader. As the boot loader is invoked at the third stage of the boot process, after the boot block, the settings in the boot loader will override the settings in the boot block.

22.6.6.1. Setting Up the Serial Console

You can easily specify the boot loader and the kernel to use the serial console by writing just one line in `/boot/loader.rc`:

```
set console="comconsole"
```

This will take effect regardless of the settings in the boot block discussed in the previous section.

You had better put the above line as the first line of `/boot/loader.rc` so as to see boot messages on the serial console as early as possible.

Likewise, you can specify the internal console as:

```
set console="vidconsole"
```

If you do not set the boot loader environment variable `console`, the boot loader, and subsequently the kernel, will use whichever console indicated by the `-h` option in the boot block.

In versions 3.2 or later, you may specify the console in `/boot/loader.conf.local` or `/boot/loader.conf`, rather than in `/boot/loader.rc`. In this method your `/boot/loader.rc` should look like:

```
include /boot/loader.4th
start
```

Then, create `/boot/loader.conf.local` and put the following line there.

```
console=comconsole
```

or

```
console=vidconsole
```

See `loader.conf(5)` for more information.

Notatka: At the moment, the boot loader has no option equivalent to the `-P` option in the boot block, and there is no provision to automatically select the internal console and the serial console based on the presence of the keyboard.

22.6.6.2. Using a Serial Port Other Than `si00` for the Console

You need to recompile the boot loader to use a serial port other than `si00` for the serial console. Follow the procedure described in Sekcja 22.6.5.2.

22.6.7. Caveats

The idea here is to allow people to set up dedicated servers that require no graphics hardware or attached keyboards. Unfortunately, while most systems will let you boot without a keyboard, there are quite a few that will not let you boot without a graphics adapter. Machines with AMI BIOSes can be configured to boot with no graphics adapter installed simply by changing the „graphics adapter” setting in the CMOS configuration to „Not installed.”

However, many machines do not support this option and will refuse to boot if you have no display hardware in the system. With these machines, you will have to leave some kind of graphics card plugged in, (even if it is just a junky mono board) although you will not have to attach a monitor. You might also try installing an AMI BIOS.

Rozdział 23. PPP and SLIP

Restructured, reorganized, and updated by Jim Mock.

23.1. Synopsis

FreeBSD has a number of ways to link one computer to another. To establish a network or Internet connection through a dial-up modem, or to allow others to do so through you, requires the use of PPP or SLIP. This chapter describes setting up these modem-based communication services in detail.

After reading this chapter, you will know:

- How to set up user PPP.
- How to set up kernel PPP.
- How to set up PPPoE (PPP over Ethernet).
- How to set up PPPoA (PPP over ATM).
- How to configure and set up a SLIP client and server.

Before reading this chapter, you should:

- Be familiar with basic network terminology.
- Understand the basics and purpose of a dialup connection and PPP and/or SLIP.

You may be wondering what the main difference is between user PPP and kernel PPP. The answer is simple: user PPP processes the inbound and outbound data in userland rather than in the kernel. This is expensive in terms of copying the data between the kernel and userland, but allows a far more feature-rich PPP implementation. User PPP uses the `tun` device to communicate with the outside world whereas kernel PPP uses the `ppp` device.

Notatka: Throughout in this chapter, user PPP will simply be referred to as **ppp** unless a distinction needs to be made between it and any other PPP software such as **pppd**. Unless otherwise stated, all of the commands explained in this chapter should be executed as `root`.

23.2. Using User PPP

Updated and enhanced by Tom Rhodes. Originally contributed by Brian Somers. With input from Nik Clayton, Dirk Frömberg, i Peter Childs.

23.2.1. User PPP

23.2.1.1. Assumptions

This document assumes you have the following:

- An account with an Internet Service Provider (ISP) which you connect to using PPP.
- You have a modem or other device connected to your system and configured correctly which allows you to connect to your ISP.
- The dial-up number(s) of your ISP.
- Your login name and password. (Either a regular UNIX style login and password pair, or a PAP or CHAP login and password pair.)
- The IP address of one or more name servers. Normally, you will be given two IP addresses by your ISP to use for this. If they have not given you at least one, then you can use the `enable dns` command in `ppp.conf` and `ppp` will set the name servers for you. This feature depends on your ISP's PPP implementation supporting DNS negotiation.

The following information may be supplied by your ISP, but is not completely necessary:

- The IP address of your ISP's gateway. The gateway is the machine to which you will connect and will be set up as your *default route*. If you do not have this information, we can make one up and your ISP's PPP server will tell us the correct value when we connect.

This IP number is referred to as `HISADDR` by `ppp`.

- The netmask you should use. If your ISP has not provided you with one, you can safely use `255.255.255.255`.
- If your ISP provides you with a static IP address and hostname, you can enter it. Otherwise, we simply let the peer assign whatever IP address it sees fit.

If you do not have any of the required information, contact your ISP.

Notatka: Throughout this section, many of the examples showing the contents of configuration files are numbered by line. These numbers serve to aid in the presentation and discussion only and are not meant to be placed in the actual file. Proper indentation with tab and space characters is also important.

23.2.1.2. Automatic PPP Configuration

Both `ppp` and `pppd` (the kernel level implementation of PPP) use the configuration files located in the `/etc/ppp` directory. Examples for user `ppp` can be found in `/usr/share/examples/ppp/`.

Configuring `ppp` requires that you edit a number of files, depending on your requirements. What you put in them depends to some extent on whether your ISP allocates IP addresses statically (i.e., you get given one IP address, and always use that one) or dynamically (i.e., your IP address changes each time you connect to your ISP).

23.2.1.2.1. PPP and Static IP Addresses

You will need to edit the `/etc/ppp/ppp.conf` configuration file. It should look similar to the example below.

Notatka: Lines that end in a `:` start in the first column (beginning of the line)— all other lines should be indented as shown using spaces or tabs.

```
1      default:
```

```

2      set log Phase Chat LCP IPCP CCP tun command
3      ident user-ppp VERSION (built COMPILATIONDATE)
4      set device /dev/cuaa0
5      set speed 115200
6      set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \
7              \"\" AT OK-AT-OK ATE1Q0 OK \\dATDT\\T TIMEOUT 40 CONNECT"
8      set timeout 180
9      enable dns
10
11  provider:
12      set phone "(123) 456 7890"
13      set authname foo
14      set authkey bar
15      set login "TIMEOUT 10 \"\" \"\" gin:--gin: \\U word: \\P col: ppp"
16      set timeout 300
17      set ifaddr x.x.x.x y.y.y.y 255.255.255.255 0.0.0.0
18      add default HISADDR

```

Line 1:

Identifies the default entry. Commands in this entry are executed automatically when ppp is run.

Line 2:

Enables logging parameters. When the configuration is working satisfactorily, this line should be reduced to saying

```
set log phase tun
```

in order to avoid excessive log file sizes.

Line 3:

Tells PPP how to identify itself to the peer. PPP identifies itself to the peer if it has any trouble negotiating and setting up the link, providing information that the peers administrator may find useful when investigating such problems.

Line 4:

Identifies the device to which the modem is connected. COM1 is /dev/cuaa0 and COM2 is /dev/cuaa1.

Line 5:

Sets the speed you want to connect at. If 115200 does not work (it should with any reasonably new modem), try 38400 instead.

Line 6 & 7:

The dial string. User PPP uses an expect-send syntax similar to the chat(8) program. Refer to the manual page for information on the features of this language.

Note that this command continues onto the next line for readability. Any command in `ppp.conf` may do this if the last character on the line is a “\” character.

Line 8:

Sets the idle timeout for the link. 180 seconds is the default, so this line is purely cosmetic.

Line 9:

Tells PPP to ask the peer to confirm the local resolver settings. If you run a local name server, this line should be commented out or removed.

Line 10:

A blank line for readability. Blank lines are ignored by PPP.

Line 11:

Identifies an entry for a provider called „provider”. This could be changed to the name of your ISP so that later you can use the `load ISP` to start the connection.

Line 12:

Sets the phone number for this provider. Multiple phone numbers may be specified using the colon (:) or pipe character (|) as a separator. The difference between the two separators is described in `ppp(8)`. To summarize, if you want to rotate through the numbers, use a colon. If you want to always attempt to dial the first number first and only use the other numbers if the first number fails, use the pipe character. Always quote the entire set of phone numbers as shown.

You must enclose the phone number in quotation marks (") if there is any intention on using spaces in the phone number. This can cause a simple, yet subtle error.

Line 13 & 14:

Identifies the user name and password. When connecting using a UNIX style login prompt, these values are referred to by the `set login` command using the `\U` and `\P` variables. When connecting using PAP or CHAP, these values are used at authentication time.

Line 15:

If you are using PAP or CHAP, there will be no login at this point, and this line should be commented out or removed. See PAP and CHAP authentication for further details.

The login string is of the same chat-like syntax as the dial string. In this example, the string works for a service whose login session looks like this:

```
J. Random Provider
login: foo
password: bar
protocol: ppp
```

You will need to alter this script to suit your own needs. When you write this script for the first time, you should ensure that you have enabled „chat” logging so you can determine if the conversation is going as expected.

Line 16:

Sets the default idle timeout (in seconds) for the connection. Here, the connection will be closed automatically after 300 seconds of inactivity. If you never want to timeout, set this value to zero or use the `-ddial` command line switch.

Line 17:

Sets the interface addresses. The string `x.x.x.x` should be replaced by the IP address that your provider has allocated to you. The string `y.y.y.y` should be replaced by the IP address that your ISP indicated for their gateway (the machine to which you connect). If your ISP has not given you a gateway address, use `10.0.0.2/0`. If you need to use a „guessed” address, make sure that you create an entry in `/etc/ppp/ppp.linkup` as per the instructions for PPP and Dynamic IP addresses. If this line is omitted, `ppp` cannot run in `-auto` mode.

Line 18:

Adds a default route to your ISP’s gateway. The special word `HISADDR` is replaced with the gateway address specified on line 17. It is important that this line appears after line 17, otherwise `HISADDR` will not yet be initialized.

If you do not wish to run `ppp` in `-auto`, this line should be moved to the `ppp.linkup` file.

It is not necessary to add an entry to `ppp.linkup` when you have a static IP address and are running `ppp` in `-auto` mode as your routing table entries are already correct before you connect. You may however wish to create an entry to invoke programs after connection. This is explained later with the `sendmail` example.

Example configuration files can be found in the `/usr/share/examples/ppp/` directory.

23.2.1.2.2. PPP and Dynamic IP Addresses

If your service provider does not assign static IP addresses, `ppp` can be configured to negotiate the local and remote addresses. This is done by „guessing” an IP address and allowing `ppp` to set it up correctly using the IP Configuration Protocol (IPCP) after connecting. The `ppp.conf` configuration is the same as PPP and Static IP Addresses, with the following change:

```
17      set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255
```

Again, do not include the line number, it is just for reference. Indentation of at least one space is required.

Line 17:

The number after the `/` character is the number of bits of the address that `ppp` will insist on. You may wish to use IP numbers more appropriate to your circumstances, but the above example will always work.

The last argument (`0.0.0.0`) tells PPP to start negotiations using address `0.0.0.0` rather than `10.0.0.1` and is necessary for some ISPs. Do not use `0.0.0.0` as the first argument to `set ifaddr` as it prevents PPP from setting up an initial route in `-auto` mode.

If you are not running in `-auto` mode, you will need to create an entry in `/etc/ppp/ppp.linkup`. `ppp.linkup` is used after a connection has been established. At this point, `ppp` will have assigned the interface addresses and it will now be possible to add the routing table entries:

```
1      provider:
2      add default HISADDR
```

Line 1:

On establishing a connection, `ppp` will look for an entry in `ppp.linkup` according to the following rules: First, try to match the same label as we used in `ppp.conf`. If that fails, look for an entry for the IP address of our gateway. This entry is a four-octet IP style label. If we still have not found an entry, look for the `MYADDR` entry.

Line 2:

This line tells `ppp` to add a default route that points to `HISADDR`. `HISADDR` will be replaced with the IP number of the gateway as negotiated by the IPCP.

See the `pmdemand` entry in the files `/usr/share/examples/ppp/ppp.conf.sample` and `/usr/share/examples/ppp/ppp.linkup.sample` for a detailed example.

23.2.1.2.3. Receiving Incoming Calls

When you configure **ppp** to receive incoming calls on a machine connected to a LAN, you must decide if you wish to forward packets to the LAN. If you do, you should allocate the peer an IP number from your LAN's subnet, and use the command `enable proxy` in your `/etc/ppp/ppp.conf` file. You should also confirm that the `/etc/rc.conf` file contains the following:

```
gateway_enable="YES"
```

23.2.1.2.4. Which getty?

Configuring FreeBSD for Dial-up Services provides a good description on enabling dial-up services using `getty(8)`.

An alternative to `getty` is `mgetty` (<http://www.leo.org/~doering/mgetty/index.html>), a smarter version of `getty` designed with dial-up lines in mind.

The advantages of using `mgetty` is that it actively *talks* to modems, meaning if port is turned off in `/etc/ttys` then your modem will not answer the phone.

Later versions of `mgetty` (from 0.99beta onwards) also support the automatic detection of PPP streams, allowing your clients script-less access to your server.

Refer to `Mgetty` and `AutoPPP` for more information on `mgetty`.

23.2.1.2.5. PPP Permissions

The `ppp` command must normally be run as the `root` user. If however, you wish to allow `ppp` to run in server mode as a normal user by executing `ppp` as described below, that user must be given permission to run `ppp` by adding them to the `network` group in `/etc/group`.

You will also need to give them access to one or more sections of the configuration file using the `allow` command:

```
allow users fred mary
```

If this command is used in the `default` section, it gives the specified users access to everything.

23.2.1.2.6. PPP Shells for Dynamic-IP Users

Create a file called `/etc/ppp/ppp-shell` containing the following:

```
#!/bin/sh
IDENT=`echo $0 | sed -e 's/^.*-\(.*\)$/\1/'`
CALLEDAS="$IDENT"
TTY=`tty`

if [ x$IDENT = xdialup ]; then
    IDENT=`basename $TTY`
fi

echo "PPP for $CALLEDAS on $TTY"
echo "Starting PPP for $IDENT"

exec /usr/sbin/ppp -direct $IDENT
```

This script should be executable. Now make a symbolic link called `ppp-dialup` to this script using the following commands:

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

You should use this script as the *shell* for all of your dialup users. This is an example from `/etc/passwd` for a dialup PPP user with username `pchilds` (remember do not directly edit the password file, use `vipw(8)`).

```
pchilds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

Create a `/home/ppp` directory that is world readable containing the following 0 byte files:

```
-r--r--r--  1 root      wheel          0 May 27 02:23 .hushlogin
-r--r--r--  1 root      wheel          0 May 27 02:22 .rhosts
```

which prevents `/etc/motd` from being displayed.

23.2.1.2.7. PPP Shells for Static-IP Users

Create the `ppp-shell` file as above, and for each account with statically assigned IPs create a symbolic link to `ppp-shell`.

For example, if you have three dialup customers, `fred`, `sam`, and `mary`, that you route class C networks for, you would type the following:

```
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-fred
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-sam
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-mary
```

Each of these users dialup accounts should have their shell set to the symbolic link created above (for example, `mary`'s shell should be `/etc/ppp/ppp-mary`).

23.2.1.2.8. Setting Up *ppp.conf* for Dynamic-IP Users

The `/etc/ppp/ppp.conf` file should contain something along the lines of:

```
default:
    set debug phase lcp chat
    set timeout 0

ttyd0:
    set ifaddr 203.14.100.1 203.14.100.20 255.255.255.255
    enable proxy

ttyd1:
    set ifaddr 203.14.100.1 203.14.100.21 255.255.255.255
    enable proxy
```

Notatka: The indenting is important.

The `default:` section is loaded for each session. For each dialup line enabled in `/etc/ttys` create an entry similar to the one for `ttyd0:` above. Each line should get a unique IP address from your pool of IP addresses for dynamic users.

23.2.1.2.9. Setting Up *ppp.conf* for Static-IP Users

Along with the contents of the sample `/usr/share/examples/ppp/ppp.conf` above you should add a section for each of the statically assigned dialup users. We will continue with our `fred`, `sam`, and `mary` example.

```
fred:
    set ifaddr 203.14.100.1 203.14.101.1 255.255.255.255

sam:
    set ifaddr 203.14.100.1 203.14.102.1 255.255.255.255

mary:
    set ifaddr 203.14.100.1 203.14.103.1 255.255.255.255
```

The file `/etc/ppp/ppp.linkup` should also contain routing information for each static IP user if required. The line below would add a route for the `203.14.101.0` class C via the client's ppp link.

```
fred:
    add 203.14.101.0 netmask 255.255.255.0 HISADDR

sam:
    add 203.14.102.0 netmask 255.255.255.0 HISADDR

mary:
    add 203.14.103.0 netmask 255.255.255.0 HISADDR
```

23.2.1.2.10. *mgetty* and AutoPPP

Configuring and compiling *mgetty* with the `AUTO_PPP` option enabled allows *mgetty* to detect the LCP phase of PPP connections and automatically spawn off a *ppp* shell. However, since the default login/password sequence does not occur it is necessary to authenticate users using either PAP or CHAP.

This section assumes the user has successfully configured, compiled, and installed a version of *mgetty* with the `AUTO_PPP` option (v0.99beta or later).

Make sure your `/usr/local/etc/mgetty+sendfax/login.config` file has the following in it:

```
/AutoPPP/ - - /etc/ppp/ppp-pap-dialup
```

This will tell *mgetty* to run the *ppp-pap-dialup* script for detected PPP connections.

Create a file called `/etc/ppp/ppp-pap-dialup` containing the following (the file should be executable):

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

For each dialup line enabled in `/etc/ttys`, create a corresponding entry in `/etc/ppp/ppp.conf`. This will happily co-exist with the definitions we created above.

```
pap:
    enable pap
    set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
    enable proxy
```

Each user logging in with this method will need to have a username/password in `/etc/ppp/ppp.secret` file, or alternatively add the following option to authenticate users via PAP from the `/etc/passwd` file.

```
enable passwdauth
```

If you wish to assign some users a static IP number, you can specify the number as the third argument in `/etc/ppp/ppp.secret`. See `/usr/share/examples/ppp/ppp.secret.sample` for examples.

23.2.1.2.11. *MS Extensions*

It is possible to configure PPP to supply DNS and NetBIOS nameserver addresses on demand.

To enable these extensions with PPP version 1.x, the following lines might be added to the relevant section of `/etc/ppp/ppp.conf`.

```
enable msextns
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

And for PPP version 2 and above:

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

This will tell the clients the primary and secondary name server addresses, and a NetBIOS nameserver host.

In version 2 and above, if the `set dns` line is omitted, PPP will use the values found in `/etc/resolv.conf`.

23.2.1.2.12. PAP and CHAP Authentication

Some ISPs set their system up so that the authentication part of your connection is done using either of the PAP or CHAP authentication mechanisms. If this is the case, your ISP will not give a `login:` prompt when you connect, but will start talking PPP immediately.

PAP is less secure than CHAP, but security is not normally an issue here as passwords, although being sent as plain text with PAP, are being transmitted down a serial line only. There is not much room for crackers to „eavesdrop”.

Referring back to the PPP and Static IP addresses or PPP and Dynamic IP addresses sections, the following alterations must be made:

```
13      set authname MyUserName
14      set authkey MyPassword
15      set login
```

Line 13:

This line specifies your PAP/CHAP user name. You will need to insert the correct value for *MyUserName*.

Line 14:

This line specifies your PAP/CHAP password. You will need to insert the correct value for *MyPassword*. You may want to add an additional line, such as:

```
16      accept PAP
or
16      accept CHAP
```

to make it obvious that this is the intention, but PAP and CHAP are both accepted by default.

Line 15:

Your ISP will not normally require that you log into the server if you are using PAP or CHAP. You must therefore disable your „set login” string.

23.2.1.2.13. Changing Your *ppp* Configuration on the Fly

It is possible to talk to the *ppp* program while it is running in the background, but only if a suitable diagnostic port has been set up. To do this, add the following line to your configuration:

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

This will tell PPP to listen to the specified UNIX domain socket, asking clients for the specified password before allowing access. The `%d` in the name is replaced with the `tun` device number that is in use.

Once a socket has been set up, the *pppctl*(8) program may be used in scripts that wish to manipulate the running program.

23.2.1.3. Using PPP Network Address Translation Capability

PPP has ability to use internal NAT without kernel diverting capabilities. This functionality may be enabled by the following line in `/etc/ppp/ppp.conf`:

```
nat enable yes
```

Alternatively, PPP NAT may be enabled by command-line option `-nat`. There is also `/etc/rc.conf` knob named `ppp_nat`, which is enabled by default.

If you use this feature, you may also find useful the following `/etc/ppp/ppp.conf` options to enable incoming connections forwarding:

```
nat port tcp 10.0.0.2:ftp ftp
nat port tcp 10.0.0.2:http http
```

or do not trust the outside at all

```
nat deny_incoming yes
```

23.2.1.4. Final System Configuration

You now have `ppp` configured, but there are a few more things to do before it is ready to work. They all involve editing the `/etc/rc.conf` file.

Working from the top down in this file, make sure the `hostname=` line is set, e.g.:

```
hostname="foo.example.com"
```

If your ISP has supplied you with a static IP address and name, it is probably best that you use this name as your host name.

Look for the `network_interfaces` variable. If you want to configure your system to dial your ISP on demand, make sure the `tun0` device is added to the list, otherwise remove it.

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

Notatka: The `ifconfig_tun0` variable should be empty, and a file called `/etc/start_if.tun0` should be created. This file should contain the line:

```
ppp -auto mysystem
```

This script is executed at network configuration time, starting your `ppp` daemon in automatic mode. If you have a LAN for which this machine is a gateway, you may also wish to use the `-alias` switch. Refer to the manual page for further details.

Make sure that the `router` program is set to `NO` with the following line in your `/etc/rc.conf`:

```
router_enable="NO"
```

It is important that the `routed` daemon is not started, as `routed` tends to delete the default routing table entries created by `ppp`.

It is probably worth your while ensuring that the `sendmail_flags` line does not include the `-q` option, otherwise `sendmail` will attempt to do a network lookup every now and then, possibly causing your machine to dial out. You may try:

```
sendmail_flags="-bd"
```

The downside of this is that you must force `sendmail` to re-examine the mail queue whenever the `ppp` link is up by typing:

```
# /usr/sbin/sendmail -q
```

You may wish to use the `!bg` command in `ppp.linkup` to do this automatically:

```
1 provider:
2 delete ALL
3 add 0 0 HISADDR
4 !bg sendmail -bd -q30m
```

If you do not like this, it is possible to set up a „dfilter” to block SMTP traffic. Refer to the sample files for further details.

All that is left is to reboot the machine. After rebooting, you can now either type:

```
# ppp
```

and then `dial provider` to start the PPP session, or, if you want `ppp` to establish sessions automatically when there is outbound traffic (and you have not created the `start_if.tun0` script), type:

```
# ppp -auto provider
```

23.2.1.5. Summary

To recap, the following steps are necessary when setting up `ppp` for the first time:

Client side:

1. Ensure that the `tun` device is built into your kernel.
2. Ensure that the `tunN` device file is available in the `/dev` directory.
3. Create an entry in `/etc/ppp/ppp.conf`. The `pmdemand` example should suffice for most ISPs.
4. If you have a dynamic IP address, create an entry in `/etc/ppp/ppp.linkup`.
5. Update your `/etc/rc.conf` file.
6. Create a `start_if.tun0` script if you require demand dialing.

Server side:

1. Ensure that the `tun` device is built into your kernel.

2. Ensure that the `tunN` device file is available in the `/dev` directory.
3. Create an entry in `/etc/passwd` (using the `vipw(8)` program).
4. Create a profile in this users home directory that runs `ppp -direct direct-server` or similar.
5. Create an entry in `/etc/ppp/ppp.conf`. The `direct-server` example should suffice.
6. Create an entry in `/etc/ppp/ppp.linkup`.
7. Update your `/etc/rc.conf` file.

23.3. Using Kernel PPP

Parts originally contributed by Gennady B. Sorokopud i Robert Huff.

23.3.1. Setting Up Kernel PPP

Before you start setting up PPP on your machine, make sure that `pppd` is located in `/usr/sbin` and the directory `/etc/ppp` exists.

`pppd` can work in two modes:

1. As a „client” — you want to connect your machine to the outside world via a PPP serial connection or modem line.
2. As a „server” — your machine is located on the network, and is used to connect other computers using PPP.

In both cases you will need to set up an options file (`/etc/ppp/options` or `~/.ppprc` if you have more than one user on your machine that uses PPP).

You will also need some modem/serial software (preferably `comms/kermit`), so you can dial and establish a connection with the remote host.

23.3.2. Using `pppd` as a Client

Based on information provided by Trev Roydhouse.

The following `/etc/ppp/options` might be used to connect to a Cisco terminal server PPP line.

```
crtstcts      # enable hardware flow control
modem         # modem controlline
noipdefault   # remote PPP server must supply your IP address
              # if the remote host does not send your IP during IPCP
              # negotiation, remove this option
passive       # wait for LCP packets
domain ppp.foo.com      # put your domain name here

:<remote_ip>    # put the IP of remote PPP host here
              # it will be used to route packets via PPP link
              # if you didn't specified the noipdefault option
```

```

        # change this line to <local_ip>:<remote_ip>

defaultroute    # put this if you want that PPP server will be your
                 # default router

```

To connect:

1. Dial to the remote host using **Kermit** (or some other modem program), and enter your user name and password (or whatever is needed to enable PPP on the remote host).
2. Exit **Kermit** (without hanging up the line).
3. Enter the following:

```
# /usr/src/usr.sbin/pppd.new/pppd /dev/tty01 19200
```

Be sure to use the appropriate speed and device name.

Now your computer is connected with PPP. If the connection fails, you can add the debug option to the `/etc/ppp/options` file, and check console messages to track the problem.

Following `/etc/ppp/pppup` script will make all 3 stages automatic:

```

#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
ps ax |grep kermit |grep -v grep
pid=`ps ax |grep kermit |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermit, PID=' ${pid}
    kill -9 ${pid}
fi

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.dial
pppd /dev/tty01 19200

```

`/etc/ppp/kermit.dial` is a **Kermit** script that dials and makes all necessary authorization on the remote host (an example of such a script is attached to the end of this document).

Use the following `/etc/ppp/pppdown` script to disconnect the PPP line:

```

#!/bin/sh
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ X${pid} != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill -TERM ${pid}
fi

ps ax |grep kermit |grep -v grep

```

```

pid=`ps ax |grep kermi |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermi, PID=' ${pid}
    kill -9 ${pid}
fi

/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
kermi -y /etc/ppp/kermi.hup
/etc/ppp/ppptest

```

Check to see if pppd is still running by executing `/usr/etc/ppp/ppptest`, which should look like this:

```

#!/bin/sh
pid=`ps ax| grep pppd |grep -v grep|awk '{print $1;}'`
if [ X${pid} != "X" ] ; then
    echo 'pppd running: PID=' ${pid-NONE}
else
    echo 'No pppd running.'
fi
set -x
netstat -n -I ppp0
ifconfig ppp0

```

To hang up the modem, execute `/etc/ppp/kermi.hup`, which should contain:

```

set line /dev/tty01      ; put your modem device here
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
echo \13
exit

```

Here is an alternate method using chat instead of kermi:

The following two files are sufficient to accomplish a pppd connection.

```

/etc/ppp/options:

/dev/cuaa1 115200

crtscts          # enable hardware flow control

```

```

modem          # modem control line
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault    # remote PPP server must supply your IP address
               # if the remote host doesn't send your IP during
               # IPCP negotiation, remove this option
passive        # wait for LCP packets
domain <your.domain> # put your domain name here

:             # put the IP of remote PPP host here
             # it will be used to route packets via PPP link
             # if you didn't specify the noipdefault option
             # change this line to <local_ip>:<remote_ip>

defaultroute   # put this if you want that PPP server will be
               # your default router

/etc/ppp/login.chat.script:

```

Notatka: The following should go on a single line.

```

ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDT<phone.number>
CONNECT "" TIMEOUT 10 ogin:-\\r-ogin: <login-id>
TIMEOUT 5 sword: <password>

```

Once these are installed and modified correctly, all you need to do is run `pppd`, like so:

```
# pppd
```

23.3.3. Using `pppd` as a Server

`/etc/ppp/options` should contain something similar to the following:

```

crtscts        # Hardware flow control
netmask 255.255.255.0 # netmask (not required)
192.114.208.20:192.114.208.165 # IP's of local and remote hosts
                                   # local ip must be different from one
                                   # you assigned to the Ethernet (or other)
                                   # interface on your machine.
                                   # remote IP is IP address that will be
                                   # assigned to the remote machine
domain ppp.foo.com # your domain
passive           # wait for LCP
modem            # modem line

```

The following `/etc/ppp/pppserv` script will tell `pppd` to behave as a server:

```

#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then

```

```

        echo 'killing pppd, PID=' ${pid}
        kill ${pid}
    fi
    ps ax |grep kermi |grep -v grep
    pid=`ps ax |grep kermi |grep -v grep|awk '{print $1;}'`
    if [ "X${pid}" != "X" ] ; then
        echo 'killing kermi, PID=' ${pid}
        kill -9 ${pid}
    fi

# reset ppp interface
ifconfig ppp0 down
ifconfig ppp0 delete

# enable autoanswer mode
kermit -y /etc/ppp/kermit.ans

# run ppp
pppd /dev/tty01 19200

```

Use this /etc/ppp/pppservdown script to stop the server:

```

#!/bin/sh
ps ax |grep pppd |grep -v grep
pid=`ps ax |grep pppd |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
ps ax |grep kermi |grep -v grep
pid=`ps ax |grep kermi |grep -v grep|awk '{print $1;}'`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermi, PID=' ${pid}
    kill -9 ${pid}
fi
ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.noans

```

The following **Kermit** script (/etc/ppp/kermit.ans) will enable/disable autoanswer mode on your modem. It should look like this:

```

set line /dev/tty01
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8

```

```

set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
inp 5 OK
echo \13
out ATS0=1\13    ; change this to out ATS0=0\13 if you want to disable
                  ; autoanswer mode

inp 5 OK
echo \13
exit

```

A script named `/etc/ppp/kermit.dial` is used for dialing and authenticating on the remote host. You will need to customize it for your needs. Put your login and password in this script; you will also need to change the input statement depending on responses from your modem and remote host.

```

;
; put the com line attached to the modem here:
;
set line /dev/tty01
;
; put the modem speed here:
;
set speed 19200
set file type binary           ; full 8 bit file xfer
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none
set modem hayes
set dial hangup off
set carrier auto               ; Then SET CARRIER if necessary,
set dial display on           ; Then SET DIAL if necessary,
set input echo on
set input timeout proceed
set input case ignore
def \%x 0                      ; login prompt counter
goto slhup

:slcmd                         ; put the modem in command mode
echo Put the modem in command mode.
clear                          ; Clear unread characters from input buffer
pause 1
output +++                     ; hayes escape sequence
input 1 OK\13\10               ; wait for OK
if success goto slhup
output \13

```

```

pause 1
output at\13
input 1 OK\13\10
if fail goto slcmd           ; if modem doesn't answer OK, try again

:slhup                        ; hang up the phone
clear                        ; Clear unread characters from input buffer
pause 1
echo Hanging up the phone.
output ath0\13               ; hayes command for on hook
input 2 OK\13\10
if fail goto slcmd           ; if no OK answer, put modem in command mode

:sldial                       ; dial the number
pause 1
echo Dialing.
output atdt9,550311\13\10    ; put phone number here
assign \%x 0                 ; zero the time counter

:look
clear                        ; Clear unread characters from input buffer
increment \%x                ; Count the seconds
input 1 {CONNECT }
if success goto sllogin
reinput 1 {NO CARRIER\13\10}
if success goto sldial
reinput 1 {NO DIALTONE\13\10}
if success goto slnodial
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 60 goto look
else goto slhup

:sllogin                      ; login
assign \%x 0                 ; zero the time counter
pause 1
echo Looking for login prompt.

:slloop
increment \%x                ; Count the seconds
clear                        ; Clear unread characters from input buffer
output \13
;
; put your expected login prompt here:
;
input 1 {Username: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup

```

```

if < \%x 10 goto slloop          ; try 10 times to get a login prompt
else goto slhup                 ; hang up and start again if 10 failures

:sluid
;
; put your userid here:
;
output ppp-login\13
input 1 {Password: }
;
; put your password here:
;
output ppp-password\13
input 1 {Entering SLIP mode.}
echo
quit

:slnodial
echo \7No dialtone.  Check the telephone line!\7
exit 1

; local variables:
; mode: csh
; comment-start: "; "
; comment-start-skip: "; "
; end:

```

23.4. Troubleshooting PPP Connections

Contributed by Tom Rhodes.

This section covers a few issues which may arise when using PPP over a modem connection. For instance, perhaps you need to know exactly what prompts the system you are dialing into will present. Some ISPs present the `ssword` prompt, and others will present `password`; if the `ppp` script is not written accordingly, the login attempt will fail. The most common way to debug `ppp` connections is by connecting manually. The following information will walk you through a manual connection step by step.

23.4.1. Check the Device Nodes

If you reconfigured your kernel then you recall the `sio` device. If you did not configure your kernel, there is no reason to worry. Just check the `dmesg` output for the modem device with:

```
# dmesg | grep sio
```

You should get some pertinent output about the `sio` devices. These are the COM ports we need. If your modem acts like a standard serial port then you should see it listed on `sio1`, or `COM2`. If so, you are not required to rebuild the kernel. When matching up `sio` modem is on `sio1` or `COM2` if you are in DOS, then your modem device would be `/dev/cuaa1`.

23.4.2. Connecting Manually

Connecting to the Internet by manually controlling `ppp` is quick, easy, and a great way to debug a connection or just get information on how your ISP treats `ppp` client connections. Let's start **PPP** from the command line. Note that in all of our examples we will use *example* as the hostname of the machine running **PPP**. You start `ppp` by just typing `ppp`:

```
# ppp
```

We have now started `ppp`.

```
ppp ON example> set device /dev/cuaa1
```

We set our modem device, in this case it is `cuaa1`.

```
ppp ON example> set speed 115200
```

Set the connection speed, in this case we are using 115,200 kbps.

```
ppp ON example> enable dns
```

Tell `ppp` to configure our resolver and add the nameserver lines to `/etc/resolv.conf`. If `ppp` cannot determine our hostname, we can set one manually later.

```
ppp ON example> term
```

Switch to „terminal” mode so that we can manually control the modem.

```
deflink: Entering terminal mode on /dev/cuaa1
type '~h' for help
```

```
at
OK
atdt123456789
```

Use `at` to initialize the modem, then use `atdt` and the number for your ISP to begin the dial in process.

```
CONNECT
```

Confirmation of the connection, if we are going to have any connection problems, unrelated to hardware, here is where we will attempt to resolve them.

```
ISP Login:myusername
```

Here you are prompted for a username, return the prompt with the username that was provided by the ISP.

```
ISP Pass:mypassword
```

This time we are prompted for a password, just reply with the password that was provided by the ISP. Just like logging into FreeBSD, the password will not echo.

```
Shell or PPP:ppp
```

Depending on your ISP this prompt may never appear. Here we are being asked if we wish to use a shell on the provider, or to start `ppp`. In this example, we have chosen to use `ppp` as we want an Internet connection.

```
Ppp ON example>
```

Notice that in this example the first `p` has been capitalized. This shows that we have successfully connected to the ISP.

```
PPp ON example>
```

We have successfully authenticated with our ISP and are waiting for the assigned IP address.

```
PPP ON example>
```

We have made an agreement on an IP address and successfully completed our connection.

```
PPP ON example>add default HISADDR
```

Here we add our default route, we need to do this before we can talk to the outside world as currently the only established connection is with the peer. If this fails due to existing routes you can put a bang character `!` in front of the `add`. Alternatively, you can set this before making the actual connection and it will negotiate a new route accordingly.

If everything went good we should now have an active connection to the Internet, which could be thrown into the background using **CTRL+Z**. If you notice the `PPP` return to `ppp` then we have lost our connection. This is good to know because it shows our connection status. Capital `P`'s show that we have a connection to the ISP and lowercase `p`'s show that the connection has been lost for whatever reason. `ppp` only has these 2 states.

23.4.2.1. Debugging

If you have a direct line and cannot seem to make a connection, then turn hardware flow CTS/RTS to off with the `set ctsrts off`. This is mainly the case if you are connected to some **PPP** capable terminal servers, where **PPP** hangs when it tries to write data to your communication link, so it would be waiting for a CTS, or Clear To Send signal which may never come. If you use this option however, you should also use the `set accmap` option, which may be required to defeat hardware dependent on passing certain characters from end to end, most of the time XON/XOFF. See the `ppp(8)` manual page for more information on this option, and how it is used.

If you have an older modem, you may need to use the `set parity even`. Parity is set at none by default, but is used for error checking (with a large increase in traffic) on older modems and some ISPs. You may need this option for the Compuserve ISP.

PPP may not return to the command mode, which is usually a negotiation error where the ISP is waiting for your side to start negotiating. At this point, using the `~p` command will force `ppp` to start sending the configuration information.

If you never obtain a login prompt, then most likely you need to use PAP or CHAP authentication instead of the UNIX style in the example above. To use PAP or CHAP just add the following options to **PPP** before going into terminal mode:

```
ppp ON example> set authname myusername
```

Where `myusername` should be replaced with the username that was assigned by the ISP.

```
ppp ON example> set authkey mypassword
```

Where `mypassword` should be replaced with the password that was assigned by the ISP.

If you connect fine, but cannot seem to find any domain name, try to use `ping(8)` with an IP address and see if you can get any return information. If you experience 100 percent (100%) packet loss, then it is most likely that you were not assigned a default route. Double check that the option `add default HISADDR` was set during the connection. If you can connect to a remote IP address then it is possible that a resolver address has not been added to the `/etc/resolv.conf`. This file should look like:

```
domain example.com
nameserver x.x.x.x
nameserver y.y.y.y
```

Where `x.x.x.x` and `y.y.y.y` should be replaced with the IP address of your ISP's DNS servers. This information may or may not have been provided when you signed up, but a quick call to your ISP should remedy that.

You could also have `syslog(3)` provide a logging function for your **PPP** connection. Just add:

```
!ppp
*. *      /var/log/ppp.log
```

to `/etc/syslog.conf`. In most cases, this functionality already exists.

23.5. Using PPP over Ethernet (PPPoE)

Contributed (from <http://node.to/freebsd/how-tos/how-to-freebsd-pppoe.html>) by Jim Mock.

This section describes how to set up PPP over Ethernet (PPPoE).

23.5.1. Configuring the Kernel

No kernel configuration is necessary for PPPoE any longer. If the necessary netgraph support is not built into the kernel, it will be dynamically loaded by `ppp`.

23.5.2. Setting Up `ppp.conf`

Here is an example of a working `ppp.conf`:

```
default:
    set log Phase tun command # you can add more detailed logging if you wish
    set ifaddr 10.0.0.1/0 10.0.0.2/0

name_of_service_provider:
    set device PPPoE:x11 # replace x11 with your Ethernet device
    set authname YOURLOGINNAME
    set authkey YOURPASSWORD
    set dial
    set login
    add default HISADDR
```

23.5.3. Running ppp

As root, you can run:

```
# ppp -ddial name_of_service_provider
```

23.5.4. Starting ppp at Boot

Add the following to your `/etc/rc.conf` file:

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_nat="YES"    # if you want to enable nat for your local network, otherwise NO
ppp_profile="name_of_service_provider"
```

23.5.5. Using a PPPoE Service Tag

Sometimes it will be necessary to use a service tag to establish your connection. Service tags are used to distinguish between different PPPoE servers attached to a given network.

You should have been given any required service tag information in the documentation provided by your ISP. If you cannot locate it there, ask your ISP's tech support personnel.

As a last resort, you could try the method suggested by the Roaring Penguin PPPoE (<http://www.roaringpenguin.com/pppoe/>) program which can be found in the Ports Collection. Bear in mind however, this may de-program your modem and render it useless, so think twice before doing it. Simply install the program shipped with the modem by your provider. Then, access the **System** menu from the program. The name of your profile should be listed there. It is usually *ISP*.

The profile name (service tag) will be used in the PPPoE configuration entry in `ppp.conf` as the provider part of the `set device` command (see the `ppp(8)` manual page for full details). It should look like this:

```
set device PPPoE:x11:ISP
```

Do not forget to change `x11` to the proper device for your Ethernet card.

Do not forget to change `ISP` to the profile you have just found above.

For additional information, see:

- Cheaper Broadband with FreeBSD on DSL (<http://renaud.waldura.com/doc/freebsd/pppoe/>) by Renaud Waldura.
- Nutzung von T-DSL und T-Online mit FreeBSD (<http://www.ruhr.de/home/nathan/FreeBSD/tdsl-freebsd.html>) by Udo Erdelhoff (in German).

23.5.6. PPPoE with a 3Com® HomeConnect® ADSL Modem Dual Link

This modem does not follow RFC 2516 (<http://www.faqs.org/rfcs/rfc2516.html>) (*A Method for transmitting PPP over Ethernet (PPPoE)*), written by L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone, and R. Wheeler). Instead, different packet type codes have been used for the Ethernet frames. Please complain to 3Com (<http://www.3com.com/>) if you think it should comply with the PPPoE specification.

In order to make FreeBSD capable of communicating with this device, a `sysctl` must be set. This can be done automatically at boot time by updating `/etc/sysctl.conf`:

```
net.graph.nonstandard_pppoe=1
```

or can be done immediately with the command:

```
# sysctl net.graph.nonstandard_pppoe=1
```

Unfortunately, because this is a system-wide setting, it is not possible to talk to a normal PPPoE client or server and a 3Com HomeConnect® ADSL Modem at the same time.

23.6. Using PPP over ATM (PPPoA)

The following describes how to set up PPP over ATM (PPPoA). PPPoA is a popular choice among European DSL providers.

23.6.1. Using PPPoA with the Alcatel SpeedTouch™ USB

PPPoA support for this device is supplied as a port in FreeBSD because the firmware is distributed under Alcatel's license agreement (http://www.speedtouchdsl.com/disclaimer_lx.htm) and can not be redistributed freely with the base system of FreeBSD.

To install the software, simply use the Ports Collection. Install the `net/pppoa` port and follow the instructions provided with it.

Like many USB devices, the Alcatel SpeedTouch™ USB needs to download firmware from the host computer to operate properly. It is possible to automate this process in FreeBSD so that this transfer takes place whenever the device is plugged into a USB port. The following information can be added to the `/etc/usbd.conf` file to enable this automatic firmware transfer. This file must be edited as the `root` user.

```
device "Alcatel SpeedTouch USB"
    devname "ugen[0-9] +"
    vendor 0x06b9
    product 0x4061
    attach "/usr/local/sbin/modem_run -f /usr/local/libdata/mgmt.o"
```

To enable the USB daemon, **usbd**, put the following the line into `/etc/rc.conf`:

```
usbd_enable="YES"
```

It is also possible to set up **ppp** to dial up at startup. To do this add the following lines to `/etc/rc.conf`. Again, for this procedure you will need to be logged in as the `root` user.

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_profile="adsl"
```

For this to work correctly you will need to have used the sample `ppp.conf` which is supplied with the `net/pppoa` port.

23.6.2. Using mpd

You can use **mpd** to connect to a variety of services, in particular PPTP services. You can find **mpd** in the Ports Collection, `net/mpd`. Many ADSL modems require that a PPTP tunnel is created between the modem and computer, one such modem is the Alcatel SpeedTouch Home.

First you must install the port, and then you can configure **mpd** to suit your requirements and provider settings. The port places a set of sample configuration files which are well documented in `PREFIX/etc/mpd/`. Note here that *PREFIX* means the directory into which your ports are installed, this defaults to `/usr/local/`. A complete guide to configure **mpd** is available in HTML format once the port has been installed. It is placed in `PREFIX/share/doc/mpd/`. Here is a sample configuration for connecting to an ADSL service with **mpd**. The configuration is spread over two files, first the `mpd.conf`:

```
default:
    load adsl

adsl:
    new -i ng0 adsl adsl
    set bundle authname username ❶
    set bundle password password ❷
    set bundle disable multilink

    set link no pap acfcomp protocomp
    set link disable chap
    set link accept chap
    set link keep-alive 30 10

    set ipcp no vjcomp
    set ipcp ranges 0.0.0.0/0 0.0.0.0/0

    set iface route default
    set iface disable on-demand
    set iface enable proxy-arp
    set iface idle 0

    open
```

❶ The username used to authenticate with your ISP.

❷ The password used to authenticate with your ISP.

The `mpd.links` file contains information about the link, or links, you wish to establish. An example `mpd.links` to accompany the above example is given beneath:

```
adsl:
    set link type pptp
    set pptp mode active
    set pptp enable originate outcall
    set pptp self 10.0.0.1 ❶
    set pptp peer 10.0.0.138 ❷
```

❶ The IP address of your FreeBSD computer which you will be using **mpd** from.

❷ The IP address of your ADSL modem. For the Alcatel SpeedTouch Home this address defaults to 10.0.0.138.

It is possible to initialize the connection easily by issuing the following command as `root`:

```
# mpd -b adsl
```

You can see the status of the connection with the following command:

```
% ifconfig ng0
ng0: flags=88d1<UP,POINTOPOINT,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
    inet 216.136.204.117 --> 204.152.186.171 netmask 0xffffffff
```

Using **mpd** is the recommended way to connect to an ADSL service with FreeBSD.

23.6.3. Using pptpclient

It is also possible to use FreeBSD to connect to other PPPoA services using `net/pptpclient`.

To use `net/pptpclient` to connect to a DSL service, install the port or package and edit your `/etc/ppp/ppp.conf`. You will need to be `root` to perform both of these operations. An example section of `ppp.conf` is given below. For further information on `ppp.conf` options consult the **ppp** manual page, `ppp(8)`.

```
adsl:
set log phase chat lcp ipcp ccp tun command
set timeout 0
enable dns
set authname username ❶
set authkey password ❷
set ifaddr 0 0
add default HISADDR
```

❶ The username of your account with the DSL provider.

❷ The password for your account.

Ostrzeżenie Because you must put your account's password in the `ppp.conf` file in plain text form you should make sure that nobody can read the contents of this file. The following series of commands will make sure the file is only readable by the `root` account. Refer to the manual pages for `chmod(1)` and `chown(8)` for further information.

```
# chown root:wheel /etc/ppp/ppp.conf
# chmod 600 /etc/ppp/ppp.conf
```

This will open a tunnel for a PPP session to your DSL router. Ethernet DSL modems have a preconfigured LAN IP address which you connect to. In the case of the Alcatel SpeedTouch Home this address is 10.0.0.138. Your router documentation should tell you which address your device uses. To open the tunnel and start a PPP session execute the following command:

```
# pptp address adsl
```

Podpowiedź: You may wish to add an ampersand („&”) to the end of the previous command because **pptp** will not return your prompt to you otherwise.

A tun virtual tunnel device will be created for interaction between the **pptp** and **ppp** processes. Once you have been returned to your prompt, or the **pptp** process has confirmed a connection you can examine the tunnel like so:

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
    inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffffff00
    Opened by PID 918
```

If you are unable to connect, check the configuration of your router, which is usually accessible via **telnet** or with a web browser. If you still cannot connect you should examine the output of the **pptp** command and the contents of the **ppp** log file, `/var/log/ppp.log` for clues.

23.7. Using SLIP

Originally contributed by Satoshi Asami. With input from Guy Helmer i Piero Serini.

23.7.1. Setting Up a SLIP Client

The following is one way to set up a FreeBSD machine for SLIP on a static host network. For dynamic hostname assignments (your address changes each time you dial up), you probably need to have a more complex setup.

First, determine which serial port your modem is connected to. Many people set up a symbolic link, such as `/dev/modem`, to point to the real device name, `/dev/cuaaN` (or `/dev/cuadN` under FreeBSD 6.X). This allows you to abstract the actual device name should you ever need to move the modem to a different port. It can become quite cumbersome when you need to fix a bunch of files in `/etc` and `.kermrc` files all over the system!

Notatka: `/dev/cuaa0` (or `/dev/cuad0` under FreeBSD 6.X) is COM1, `cuaa1` (or `/dev/cuad1`) is COM2, etc.

Make sure you have the following in your kernel configuration file:

```
device    sl
```

It is included in the `GENERIC` kernel, so this should not be a problem unless you have deleted it.

23.7.1.1. Things You Have to Do Only Once

1. Add your home machine, the gateway and nameservers to your `/etc/hosts` file. Ours looks like this:

```
127.0.0.1          localhost loghost
136.152.64.181     water.CS.Example.EDU water.CS water
136.152.64.1       inr-3.CS.Example.EDU inr-3 slip-gateway
128.32.136.9       ns1.Example.EDU ns1
128.32.136.12      ns2.Example.EDU ns2
```

2. Make sure you have `hosts` before `bind` in your `/etc/host.conf` on FreeBSD versions prior to 5.0. Since FreeBSD 5.0, the system uses the file `/etc/nsswitch.conf` instead, make sure you have `files` before `dns` in the `hosts` line of this file. Without these parameters funny things may happen.
3. Edit the `/etc/rc.conf` file.
 1. Set your hostname by editing the line that says:


```
hostname="myname.my.domain"
```

Your machine's full Internet hostname should be placed here.
 2. Designate the default router by changing the line:


```
defaultrouter="NO"
```

to:

```
defaultrouter="slip-gateway"
```
4. Make a file `/etc/resolv.conf` which contains:


```
domain CS.Example.EDU
nameserver 128.32.136.9
nameserver 128.32.136.12
```

As you can see, these set up the nameserver hosts. Of course, the actual domain names and addresses depend on your environment.
5. Set the password for `root` and `toor` (and any other accounts that do not have a password).
6. Reboot your machine and make sure it comes up with the correct hostname.

23.7.1.2. Making a SLIP Connection

1. Dial up, type `slip` at the prompt, enter your machine name and password. What is required to be entered depends on your environment. If you use **Kermit**, you can try a script like this:

```
# kermit setup
set modem hayes
set line /dev/modem
set speed 115200
set parity none
set flow rts/cts
set terminal bytesize 8
set file type binary
# The next macro will dial up and login
define slip dial 643-9600, input 10 =>, if failure stop, -
output slip\x0d, input 10 Username:, if failure stop, -
output silvia\x0d, input 10 Password:, if failure stop, -
output ***\x0d, echo \x0aCONNECTED\x0a
```

Of course, you have to change the username and password to fit yours. After doing so, you can just type `slip` from the **Kermit** prompt to connect.

Notatka: Leaving your password in plain text anywhere in the filesystem is generally a *bad* idea. Do it at your own risk.

2. Leave the **Kermit** there (you can suspend it by **Ctrl-z**) and as **root**, type:

```
# slattach -h -c -s 115200 /dev/modem
```

If you are able to ping hosts on the other side of the router, you are connected! If it does not work, you might want to try **-a** instead of **-c** as an argument to **slattach**.

23.7.1.3. How to Shutdown the Connection

Do the following:

```
# kill -INT `cat /var/run/slattach.modem.pid`
```

to kill **slattach**. Keep in mind you must be **root** to do the above. Then go back to **kermit** (by running **fg** if you suspended it) and exit from it (**q**).

The **slattach(8)** manual page says you have to use **ifconfig s10 down** to mark the interface down, but this does not seem to make any difference. (**ifconfig s10** reports the same thing.)

Some times, your modem might refuse to drop the carrier. In that case, simply start **kermit** and quit it again. It usually goes out on the second try.

23.7.1.4. Troubleshooting

If it does not work, feel free to ask on **freebsd-net** (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-net>) mailing list. The things that people tripped over so far:

- Not using **-c** or **-a** in **slattach** (This should not be fatal, but some users have reported that this solves their problems.)
- Using **s10** instead of **sl0** (might be hard to see the difference on some fonts).
- Try **ifconfig s10** to see your interface status. For example, you might get:

```
# ifconfig s10
s10: flags=10<POINTOPOINT>
    inet 136.152.64.181 --> 136.152.64.1 netmask ffffffff00
```

- If you get no route to host messages from **ping(8)**, there may be a problem with your routing table. You can use the **netstat -r** command to display the current routes :

```
# netstat -r
Routing tables
Destination      Gateway          Flags    Refs      Use  IfaceMTU    Rtt      Netmasks:

(root node)
(root node)

Route Tree for Protocol Family inet:
(root node) =>
default          inr-3.Example.EDU  UG          8    224515  s10 -        -
localhost.Exampl localhost.Example. UH          5     42127  lo0 -        0.438
inr-3.Example.ED water.CS.Example.E UH          1         0  s10 -        -
```

```
water.CS.Example localhost.Example. UGH          34 47641234 100 -          0.438
(root node)
```

The preceding examples are from a relatively busy system. The numbers on your system will vary depending on network activity.

23.7.2. Setting Up a SLIP Server

This document provides suggestions for setting up SLIP Server services on a FreeBSD system, which typically means configuring your system to automatically start up connections upon login for remote SLIP clients.

23.7.2.1. Prerequisites

This section is very technical in nature, so background knowledge is required. It is assumed that you are familiar with the TCP/IP network protocol, and in particular, network and node addressing, network address masks, subnetting, routing, and routing protocols, such as RIP. Configuring SLIP services on a dial-up server requires a knowledge of these concepts, and if you are not familiar with them, please read a copy of either Craig Hunt's *TCP/IP Network Administration* published by O'Reilly & Associates, Inc. (ISBN Number 0-937175-82-X), or Douglas Comer's books on the TCP/IP protocol.

It is further assumed that you have already set up your modem(s) and configured the appropriate system files to allow logins through your modems. If you have not prepared your system for this yet, please see Sekcja 22.4 for details on dialup services configuration. You may also want to check the manual pages for `sio(4)` for information on the serial port device driver and `tty(5)`, `gettytab(5)`, `getty(8)`, & `init(8)` for information relevant to configuring the system to accept logins on modems, and perhaps `stty(1)` for information on setting serial port parameters (such as `cllocal` for directly-connected serial interfaces).

23.7.2.2. Quick Overview

In its typical configuration, using FreeBSD as a SLIP server works as follows: a SLIP user dials up your FreeBSD SLIP Server system and logs in with a special SLIP login ID that uses `/usr/sbin/sliplogin` as the special user's shell. The `sliplogin` program browses the file `/etc/sliphome/slip.hosts` to find a matching line for the special user, and if it finds a match, connects the serial line to an available SLIP interface and then runs the shell script `/etc/sliphome/slip.login` to configure the SLIP interface.

23.7.2.2.1. An Example of a SLIP Server Login

For example, if a SLIP user ID were `Shelmerg`, `Shelmerg`'s entry in `/etc/master.passwd` would look something like this:

```
Shelmerg:password:1964:89::0:0:Guy Helmer - SLIP:/usr/users/Shelmerg:/usr/sbin/sliplogin
```

When `Shelmerg` logs in, `sliplogin` will search `/etc/sliphome/slip.hosts` for a line that had a matching user ID; for example, there may be a line in `/etc/sliphome/slip.hosts` that reads:

```
Shelmerg          dc-slip sl-helmer          0xfffffc00          autocomp
```

`sliplogin` will find that matching line, hook the serial line into the next available SLIP interface, and then execute `/etc/sliphome/slip.login` like this:

```
/etc/sliphome/slip.login 0 19200 Shelmerg dc-slip sl-helmer 0xffffffffc00 autocomp
```

If all goes well, `/etc/sliphome/slip.login` will issue an `ifconfig` for the SLIP interface to which `sliplogin` attached itself (SLIP interface 0, in the above example, which was the first parameter in the list given to `slip.login`) to set the local IP address (`dc-slip`), remote IP address (`sl-helmer`), network mask for the SLIP interface (`0xffffffffc00`), and any additional flags (`autocomp`). If something goes wrong, `sliplogin` usually logs good informational messages via the **syslogd** daemon facility, which usually logs to `/var/log/messages` (see the manual pages for `syslogd(8)` and `syslog.conf(5)` and perhaps check `/etc/syslog.conf` to see to what **syslogd** is logging and where it is logging to).

23.7.2.3. Kernel Configuration

FreeBSD's default kernel (`GENERIC`) comes with SLIP (`sl(4)`) support; in case of a custom kernel, you have to add the following line to your kernel configuration file:

```
device    sl
```

By default, your FreeBSD machine will not forward packets. If you want your FreeBSD SLIP Server to act as a router, you will have to edit the `/etc/rc.conf` file and change the setting of the `gateway_enable` variable to `YES`.

You will then need to reboot for the new settings to take effect.

Please refer to Rozdział 8 on Configuring the FreeBSD Kernel for help in reconfiguring your kernel.

23.7.2.4. Sliplogin Configuration

As mentioned earlier, there are three files in the `/etc/sliphome` directory that are part of the configuration for `/usr/sbin/sliplogin` (see `sliplogin(8)` for the actual manual page for `sliplogin`): `slip.hosts`, which defines the SLIP users and their associated IP addresses; `slip.login`, which usually just configures the SLIP interface; and (optionally) `slip.logout`, which undoes `slip.login`'s effects when the serial connection is terminated.

23.7.2.4.1. *slip.hosts* Configuration

`/etc/sliphome/slip.hosts` contains lines which have at least four items separated by whitespace:

- SLIP user's login ID
- Local address (local to the SLIP server) of the SLIP link
- Remote address of the SLIP link
- Network mask

The local and remote addresses may be host names (resolved to IP addresses by `/etc/hosts` or by the domain name service, depending on your specifications in the file `/etc/nsswitch.conf`), and the network mask may be a name that can be resolved by a lookup into `/etc/networks`. On a sample system, `/etc/sliphome/slip.hosts` looks like this:

```
#
# login local-addr      remote-addr      mask                opt1      opt2
#                               (normal,compress,noicmp)
#
```

```
Shelmerg dc-slip      sl-helmerg      0xfffffc00      autocomp
```

At the end of the line is one or more of the options:

- `normal` — no header compression
- `compress` — compress headers
- `autocomp` — compress headers if the remote end allows it
- `noicmp` — disable ICMP packets (so any „ping” packets will be dropped instead of using up your bandwidth)

Your choice of local and remote addresses for your SLIP links depends on whether you are going to dedicate a TCP/IP subnet or if you are going to use „proxy ARP” on your SLIP server (it is not „true” proxy ARP, but that is the terminology used in this section to describe it). If you are not sure which method to select or how to assign IP addresses, please refer to the TCP/IP books referenced in the SLIP Prerequisites (Sekcja 23.7.2.1) and/or consult your IP network manager.

If you are going to use a separate subnet for your SLIP clients, you will need to allocate the subnet number out of your assigned IP network number and assign each of your SLIP client’s IP numbers out of that subnet. Then, you will probably need to configure a static route to the SLIP subnet via your SLIP server on your nearest IP router.

Otherwise, if you will use the „proxy ARP” method, you will need to assign your SLIP client’s IP addresses out of your SLIP server’s Ethernet subnet, and you will also need to adjust your `/etc/sliphome/slip.login` and `/etc/sliphome/slip.logout` scripts to use `arp(8)` to manage the proxy-ARP entries in the SLIP server’s ARP table.

23.7.2.4.2. *slip.login Configuration*

The typical `/etc/sliphome/slip.login` file looks like this:

```
#!/bin/sh -
#
#      @(#)slip.login  5.1  (Berkeley)  7/1/90

#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit  ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
```

This `slip.login` file merely runs `ifconfig` for the appropriate SLIP interface with the local and remote addresses and network mask of the SLIP interface.

If you have decided to use the „proxy ARP” method (instead of using a separate subnet for your SLIP clients), your `/etc/sliphome/slip.login` file will need to look something like this:

```
#!/bin/sh -
#
#      @(#)slip.login  5.1  (Berkeley)  7/1/90

#
```

```
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
# Answer ARP requests for the SLIP client with our Ethernet addr
/usr/sbin/arp -s $5 00:11:22:33:44:55 pub
```

The additional line in this `slip.login`, `arp -s $5 00:11:22:33:44:55 pub`, creates an ARP entry in the SLIP server's ARP table. This ARP entry causes the SLIP server to respond with the SLIP server's Ethernet MAC address whenever another IP node on the Ethernet asks to speak to the SLIP client's IP address.

When using the example above, be sure to replace the Ethernet MAC address (00:11:22:33:44:55) with the MAC address of your system's Ethernet card, or your „proxy ARP” will definitely not work! You can discover your SLIP server's Ethernet MAC address by looking at the results of running `netstat -i`; the second line of the output should look something like:

```
ed0    1500    <Link>0.2.c1.28.5f.4a          191923          0    129457          0    116
```

This indicates that this particular system's Ethernet MAC address is 00:02:c1:28:5f:4a — the periods in the Ethernet MAC address given by `netstat -i` must be changed to colons and leading zeros should be added to each single-digit hexadecimal number to convert the address into the form that `arp(8)` desires; see the manual page on `arp(8)` for complete information on usage.

Notatka: When you create `/etc/sliphome/slip.login` and `/etc/sliphome/slip.logout`, the „execute” bit (i.e., `chmod 755 /etc/sliphome/slip.login /etc/sliphome/slip.logout`) must be set, or `sliplogin` will be unable to execute it.

23.7.2.4.3. `slip.logout` Configuration

`/etc/sliphome/slip.logout` is not strictly needed (unless you are implementing „proxy ARP”), but if you decide to create it, this is an example of a basic `slip.logout` script:

```
#!/bin/sh -
#
#      slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
```

If you are using „proxy ARP”, you will want to have `/etc/sliphome/slip.logout` remove the ARP entry for the SLIP client:

```
#!/bin/sh -
```

```
#
#      @(#)slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
# /sbin/ifconfig sl$1 down
# Quit answering ARP requests for the SLIP client
# /usr/sbin/arp -d $5
```

The `arp -d $5` removes the ARP entry that the „proxy ARP” `slip.login` added when the SLIP client logged in.

It bears repeating: make sure `/etc/sliphome/slip.logout` has the execute bit set after you create it (i.e., `chmod 755 /etc/sliphome/slip.logout`).

23.7.2.5. Routing Considerations

If you are not using the „proxy ARP” method for routing packets between your SLIP clients and the rest of your network (and perhaps the Internet), you will probably have to add static routes to your closest default router(s) to route your SLIP clients subnet via your SLIP server.

23.7.2.5.1. Static Routes

Adding static routes to your nearest default routers can be troublesome (or impossible if you do not have authority to do so...). If you have a multiple-router network in your organization, some routers, such as those made by Cisco and Proteon, may not only need to be configured with the static route to the SLIP subnet, but also need to be told which static routes to tell other routers about, so some expertise and troubleshooting/tweaking may be necessary to get static-route-based routing to work.

23.7.2.5.2. Running **GateD**

Notatka: **GateD** is proprietary software now and will not be available as source code to the public anymore (more info on the GateD (<http://www.gated.org/>) website). This section only exists to ensure backwards compatibility for those that are still using an older version.

An alternative to the headaches of static routes is to install **GateD** on your FreeBSD SLIP server and configure it to use the appropriate routing protocols (RIP/OSPF/BGP/EGP) to tell other routers about your SLIP subnet. You will need to write a `/etc/gated.conf` file to configure your **GateD**; here is a sample, similar to what the author used on a FreeBSD SLIP server:

```
#
# gated configuration file for dc.dsu.edu; for gated version 3.5alpha5
# Only broadcast RIP information for xxx.xxx.yy out the ed Ethernet interface
#
#
```

```

# tracing options
#
traceoptions "/var/tmp/gated.output" replace size 100k files 2 general ;

rip yes {
    interface sl noripout noripin ;
    interface ed ripin ripout version 1 ;
    traceoptions route ;
} ;

#
# Turn on a bunch of tracing info for the interface to the kernel:
kernel {
    traceoptions remnants request routes info interface ;
} ;

#
# Propagate the route to xxx.xxx.yy out the Ethernet interface via RIP
#

export proto rip interface ed {
    proto direct {
        xxx.xxx.yy mask 255.255.252.0 metric 1; # SLIP connections
    } ;
} ;

#
# Accept routes from RIP via ed Ethernet interfaces

import proto rip interface ed {
    all ;
} ;

```

The above sample `gated.conf` file broadcasts routing information regarding the SLIP subnet `xxx.xxx.yy` via RIP onto the Ethernet; if you are using a different Ethernet driver than the `ed` driver, you will need to change the references to the `ed` interface appropriately. This sample file also sets up tracing to `/var/tmp/gated.output` for debugging **GateD**'s activity; you can certainly turn off the tracing options if **GateD** works correctly for you. You will need to change the `xxx.xxx.yy`'s into the network address of your own SLIP subnet (be sure to change the net mask in the `proto direct` clause as well).

Once you have installed and configured **GateD** on your system, you will need to tell the FreeBSD startup scripts to run **GateD** in place of **routed**. The easiest way to accomplish this is to set the `router` and `router_flags` variables in `/etc/rc.conf`. Please see the manual page for **GateD** for information on command-line parameters.

Rozdział 24. Electronic Mail

Original work by Bill Lloyd. Rewritten by Jim Mock.

24.1. Synopsis

„Electronic Mail”, better known as email, is one of the most widely used forms of communication today. This chapter provides a basic introduction to running a mail server on FreeBSD, as well as an introduction to sending and receiving email using FreeBSD; however, it is not a complete reference and in fact many important considerations are omitted. For more complete coverage of the subject, the reader is referred to the many excellent books listed in Dodatek B.

After reading this chapter, you will know:

- What software components are involved in sending and receiving electronic mail.
- Where basic **sendmail** configuration files are located in FreeBSD.
- The difference between remote and local mailboxes.
- How to block spammers from illegally using your mail server as a relay.
- How to install and configure an alternate Mail Transfer Agent on your system, replacing **sendmail**.
- How to troubleshoot common mail server problems.
- How to use SMTP with UUCP.
- How to set up the system to send mail only.
- How to use mail with a dialup connection.
- How to configure SMTP Authentication for added security.
- How to install and use a Mail User Agent, such as **mutt** to send and receive email.
- How to download your mail from a remote POP or IMAP server.
- How to automatically apply filters and rules to incoming email.

Before reading this chapter, you should:

- Properly set up your network connection (Rozdział 27).
- Properly set up the DNS information for your mail host (Rozdział 25).
- Know how to install additional third-party software (Rozdział 4).

24.2. Using Electronic Mail

There are five major parts involved in an email exchange. They are: the user program, the server daemon, DNS, a remote or local mailbox, and of course, the mailhost itself.

24.2.1. The User Program

This includes command line programs such as **mutt**, **pine**, **elm**, and **mail**, and GUI programs such as **balsa**, **xfmail** to name a few, and something more „sophisticated” like a WWW browser. These programs simply pass off the email transactions to the local „mailhost”, either by calling one of the server daemons available, or delivering it over TCP.

24.2.2. Mailhost Server Daemon

FreeBSD ships with **sendmail** by default, but also support numerous other mail server daemons, just some of which include:

- **exim**;
- **postfix**;
- **qmail**.

The server daemon usually has two functions—it is responsible for receiving incoming mail as well as delivering outgoing mail. It is *not* responsible for the collection of mail using protocols such as POP or IMAP to read your email, nor does it allow connecting to local **mbox** or **Maildir** mailboxes. You may require an additional daemon for that.

Ostrzeżenie Older versions of **sendmail** have some serious security issues which may result in an attacker gaining local and/or remote access to your machine. Make sure that you are running a current version to avoid these problems. Optionally, install an alternative MTA from the **FreeBSD Ports Collection**.

24.2.3. Email and DNS

The Domain Name System (DNS) and its daemon **named** play a large role in the delivery of email. In order to deliver mail from your site to another, the server daemon will look up the remote site in the DNS to determine the host that will receive mail for the destination. This process also occurs when mail is sent from a remote host to your mail server.

DNS is responsible for mapping hostnames to IP addresses, as well as for storing information specific to mail delivery, known as MX records. The MX (Mail eXchanger) record specifies which host, or hosts, will receive mail for a particular domain. If you do not have an MX record for your hostname or domain, the mail will be delivered directly to your host provided you have an A record pointing your hostname to your IP address.

You may view the MX records for any domain by using the **host(1)** command, as seen in the example below:

```
% host -t mx FreeBSD.org
FreeBSD.org mail is handled (pri=10) by mx1.FreeBSD.org
```

24.2.4. Receiving Mail

Receiving mail for your domain is done by the mail host. It will collect all mail sent to your domain and store it either in **mbox** (the default method for storing mail) or **Maildir** format, depending on your configuration. Once mail has been stored, it may either be read locally using applications such as **mail(1)** or **mutt**, or remotely accessed and

collected using protocols such as POP or IMAP. This means that should you only wish to read mail locally, you are not required to install a POP or IMAP server.

24.2.4.1. Accessing remote mailboxes using POP and IMAP

In order to access mailboxes remotely, you are required to have access to a POP or IMAP server. These protocols allow users to connect to their mailboxes from remote locations with ease. Though both POP and IMAP allow users to remotely access mailboxes, IMAP offers many advantages, some of which are:

- IMAP can store messages on a remote server as well as fetch them.
- IMAP supports concurrent updates.
- IMAP can be extremely useful over low-speed links as it allows users to fetch the structure of messages without downloading them; it can also perform tasks such as searching on the server in order to minimize data transfer between clients and servers.

In order to install a POP or IMAP server, the following steps should be performed:

1. Choose an IMAP or POP server that best suits your needs. The following POP and IMAP servers are well known and serve as some good examples:
 - **qpopper**;
 - **teapop**;
 - **imap-uw**;
 - **courier-imap**;
2. Install the POP or IMAP daemon of your choosing from the ports collection.
3. Where required, modify `/etc/inetd.conf` to load the POP or IMAP server.

Ostrzeżenie! It should be noted that both POP and IMAP transmit information, including username and password credentials in clear-text. This means that if you wish to secure the transmission of information across these protocols, you should consider tunneling sessions over `ssh(1)`. Tunneling sessions is described in *Sekcja 14.11.8*.

24.2.4.2. Accessing local mailboxes

Mailboxes may be accessed locally by directly utilizing MUAs on the server on which the mailbox resides. This can be done using applications such as **mutt** or `mail(1)`.

24.2.5. The Mail Host

The mail host is the name given to a server that is responsible for delivering and receiving mail for your host, and possibly your network.

24.3. sendmail Configuration

Contributed by Christopher Shumway.

sendmail(8) is the default Mail Transfer Agent (MTA) in FreeBSD. **sendmail**'s job is to accept mail from Mail User Agents (MUA) and deliver it to the appropriate mailer as defined by its configuration file. **sendmail** can also accept network connections and deliver mail to local mailboxes or deliver it to another program.

sendmail uses the following configuration files:

Filename	Function
/etc/mail/access	sendmail access database file
/etc/mail/aliases	Mailbox aliases
/etc/mail/local-host-names	Lists of hosts sendmail accepts mail for
/etc/mail/mailer.conf	Mailer program configuration
/etc/mail/mailertable	Mailer delivery table
/etc/mail/sendmail.cf	sendmail master configuration file
/etc/mail/virtusertable	Virtual users and domain tables

24.3.1. /etc/mail/access

The access database defines what host(s) or IP addresses have access to the local mail server and what kind of access they have. Hosts can be listed as OK, REJECT, RELAY or simply passed to **sendmail**'s error handling routine with a given mailer error. Hosts that are listed as OK, which is the default, are allowed to send mail to this host as long as the mail's final destination is the local machine. Hosts that are listed as REJECT are rejected for all mail connections. Hosts that have the RELAY option for their hostname are allowed to send mail for any destination through this mail server.

Przykład 24-1. Configuring the sendmail Access Database

cyberspammer.com	550 We do not accept mail from spammers
FREE.STEALTH.MAILER@	550 We do not accept mail from spammers
another.source.of.spam	REJECT
okay.cyberspammer.com	OK
128.32	RELAY

In this example we have five entries. Mail senders that match the left hand side of the table are affected by the action on the right side of the table. The first two examples give an error code to **sendmail**'s error handling routine. The message is printed to the remote host when a mail matches the left hand side of the table. The next entry rejects mail from a specific host on the Internet, `another.source.of.spam`. The next entry accepts mail connections from a host `okay.cyberspammer.com`, which is more exact than the `cyberspammer.com` line above. More specific matches override less exact matches. The last entry allows relaying of electronic mail from hosts with an IP address that begins with `128.32`. These hosts would be able to send mail through this mail server that are destined for other mail servers.

When this file is updated, you need to run `make` in `/etc/mail/` to update the database.

24.3.2. /etc/mail/aliases

The aliases database contains a list of virtual mailboxes that are expanded to other user(s), files, programs or other aliases. Here are a few examples that can be used in `/etc/mail/aliases`:

Przykład 24-2. Mail Aliases

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket: /dev/null
procmail: "|/usr/local/bin/procmail"
```

The file format is simple; the mailbox name on the left side of the colon is expanded to the target(s) on the right. The first example simply expands the mailbox `root` to the mailbox `localuser`, which is then looked up again in the aliases database. If no match is found, then the message is delivered to the local user `localuser`. The next example shows a mail list. Mail to the mailbox `ftp-bugs` is expanded to the three local mailboxes `joe`, `eric`, and `paul`. Note that a remote mailbox could be specified as `<user@example.com>`. The next example shows writing mail to a file, in this case `/dev/null`. The last example shows sending mail to a program, in this case the mail message is written to the standard input of `/usr/local/bin/procmail` through a UNIX pipe.

When this file is updated, you need to run `make` in `/etc/mail/` to update the database.

24.3.3. /etc/mail/local-host-names

This is a list of hostnames `sendmail(8)` is to accept as the local host name. Place any domains or hosts that **sendmail** is to be receiving mail for. For example, if this mail server was to accept mail for the domain `example.com` and the host `mail.example.com`, its `local-host-names` might look something like this:

```
example.com
mail.example.com
```

When this file is updated, `sendmail(8)` needs to be restarted to read the changes.

24.3.4. /etc/mail/sendmail.cf

sendmail's master configuration file, `sendmail.cf` controls the overall behavior of **sendmail**, including everything from rewriting e-mail addresses to printing rejection messages to remote mail servers. Naturally, with such a diverse role, this configuration file is quite complex and its details are a bit out of the scope of this section. Fortunately, this file rarely needs to be changed for standard mail servers.

The master **sendmail** configuration file can be built from `m4(1)` macros that define the features and behavior of **sendmail**. Please see `/usr/src/contrib/sendmail/cf/README` for some of the details.

When changes to this file are made, **sendmail** needs to be restarted for the changes to take effect.

24.3.5. /etc/mail/virtusertable

The `virtusertable` maps mail addresses for virtual domains and mailboxes to real mailboxes. These mailboxes can be local, remote, aliases defined in `/etc/mail/aliases` or files.

Przykład 24-3. Example Virtual Domain Mail Map

root@example.com	root
postmaster@example.com	postmaster@noc.example.net
@example.com	joe

In the above example, we have a mapping for a domain `example.com`. This file is processed in a first match order down the file. The first item maps `<root@example.com>` to the local mailbox `root`. The next entry maps `<postmaster@example.com>` to the mailbox `postmaster` on the host `noc.example.net`. Finally, if nothing from `example.com` has matched so far, it will match the last mapping, which matches every other mail message addressed to someone at `example.com`. This will be mapped to the local mailbox `joe`.

24.4. Changing Your Mail Transfer Agent

Written by Andrew Boothman. Information taken from e-mails written by Gregory Neil Shapiro.

As already mentioned, FreeBSD comes with **sendmail** already installed as your MTA (Mail Transfer Agent). Therefore by default it is in charge of your outgoing and incoming mail.

However, for a variety of reasons, some system administrators want to change their system's MTA. These reasons range from simply wanting to try out another MTA to needing a specific feature or package which relies on another mailer. Fortunately, whatever the reason, FreeBSD makes it easy to make the change.

24.4.1. Install a New MTA

You have a wide choice of MTAs available. A good starting point is the FreeBSD Ports Collection where you will be able to find many. Of course you are free to use any MTA you want from any location, as long as you can make it run under FreeBSD.

Start by installing your new MTA. Once it is installed it gives you a chance to decide if it really fulfills your needs, and also gives you the opportunity to configure your new software before getting it to take over from **sendmail**.

When doing this, you should be sure that installing the new software will not attempt to overwrite system binaries such as `/usr/bin/sendmail`. Otherwise, your new mail software has essentially been put into service before you have configured it.

Please refer to your chosen MTA's documentation for information on how to configure the software you have chosen.

24.4.2. Disable sendmail

The procedure used to start **sendmail** changed significantly between 4.5-RELEASE, 4.6-RELEASE, and later releases. Therefore, the procedure used to disable it is subtly different.

Ostrzeżenie If you disable **sendmail**'s outgoing mail service, it is important that you replace it with an alternative mail delivery system. If you choose not to, system functions such as `periodic(8)` will be unable to deliver their results by e-mail as they would normally expect to. Many parts of your system may expect to have a functional **sendmail**-compatible system. If applications continue to use **sendmail**'s binaries to try to send e-mail after you have disabled them, mail could go into an inactive **sendmail** queue, and never be delivered.

24.4.2.1. FreeBSD 4.5-STABLE before 2002/4/4 and Earlier (Including 4.5-RELEASE and Earlier)

Enter:

```
sendmail_enable="NO"
```

into `/etc/rc.conf`. This will disable **sendmail**'s incoming mail service, but if `/etc/mail/mailler.conf` (see below) is not changed, **sendmail** will still be used to send e-mail.

24.4.2.2. FreeBSD 4.5-STABLE after 2002/4/4 (Including 4.6-RELEASE and Later)

In order to completely disable **sendmail**, including the outgoing mail service, you must use

```
sendmail_enable="NONE"
```

```
in /etc/rc.conf.
```

If you only want to disable **sendmail**'s incoming mail service, you should set

```
sendmail_enable="NO"
```

in `/etc/rc.conf`. However, if incoming mail is disabled, local delivery will still function. More information on **sendmail**'s startup options is available from the `rc.sendmail(8)` manual page.

24.4.2.3. FreeBSD 5.0-STABLE and Later

In order to completely disable **sendmail**, including the outgoing mail service, you must use

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
```

```
in /etc/rc.conf.
```

If you only want to disable **sendmail**'s incoming mail service, you should set

```
sendmail_enable="NO"
```

in `/etc/rc.conf`. More information on **sendmail**'s startup options is available from the `rc.sendmail(8)` manual page.

24.4.3. Running Your New MTA on Boot

You may have a choice of two methods for running your new MTA on boot, again depending on what version of FreeBSD you are running.

24.4.3.1. FreeBSD 4.5-STABLE before 2002/4/11 (Including 4.5-RELEASE and Earlier)

Add a script to `/usr/local/etc/rc.d/` that ends in `.sh` and is executable by `root`. The script should accept `start` and `stop` parameters. At startup time the system scripts will execute the command

```
/usr/local/etc/rc.d/supermailer.sh start
```

which you can also use to manually start the server. At shutdown time, the system scripts will use the `stop` option, running the command

```
/usr/local/etc/rc.d/supermailer.sh stop
```

which you can also use to manually stop the server while the system is running.

24.4.3.2. FreeBSD 4.5-STABLE after 2002/4/11 (Including 4.6-RELEASE and Later)

With later versions of FreeBSD, you can use the above method or you can set

```
mta_start_script="filename"
```

in `/etc/rc.conf`, where *filename* is the name of some script that you want executed at boot to start your MTA.

24.4.4. Replacing sendmail as the System's Default Mailer

The program **sendmail** is so ubiquitous as standard software on UNIX systems that some software just assumes it is already installed and configured. For this reason, many alternative MTA's provide their own compatible implementations of the **sendmail** command-line interface; this facilitates using them as „drop-in” replacements for **sendmail**.

Therefore, if you are using an alternative mailer, you will need to make sure that software trying to execute standard **sendmail** binaries such as `/usr/bin/sendmail` actually executes your chosen mailer instead. Fortunately, FreeBSD provides a system called `mailwrapper(8)` that does this job for you.

When **sendmail** is operating as installed, you will find something like the following in `/etc/mail/mailer.conf`:

```
sendmail      /usr/libexec/sendmail/sendmail
send-mail     /usr/libexec/sendmail/sendmail
mailq         /usr/libexec/sendmail/sendmail
newaliases   /usr/libexec/sendmail/sendmail
hoststat      /usr/libexec/sendmail/sendmail
purgestat     /usr/libexec/sendmail/sendmail
```

This means that when any of these common commands (such as `sendmail` itself) are run, the system actually invokes a copy of `mailwrapper` named `sendmail`, which checks `mailer.conf` and executes `/usr/libexec/sendmail/sendmail` instead. This system makes it easy to change what binaries are actually executed when these default `sendmail` functions are invoked.

Therefore if you wanted `/usr/local/supermailer/bin/sendmail-compat` to be run instead of **sendmail**, you could change `/etc/mail/mailer.conf` to read:

```
sendmail      /usr/local/supermailer/bin/sendmail-compat
send-mail     /usr/local/supermailer/bin/sendmail-compat
mailq         /usr/local/supermailer/bin/mailq-compat
newaliases   /usr/local/supermailer/bin/newaliases-compat
hoststat      /usr/local/supermailer/bin/hoststat-compat
purgestat     /usr/local/supermailer/bin/purgestat-compat
```

24.4.5. Finishing

Once you have everything configured the way you want it, you should either kill the **sendmail** processes that you no longer need and start the processes belonging to your new software, or simply reboot. Rebooting will also give you the opportunity to ensure that you have correctly configured your system to start your new MTA automatically on boot.

24.5. Troubleshooting

1. Why do I have to use the FQDN for hosts on my site?

You will probably find that the host is actually in a different domain; for example, if you are in `foo.bar.edu` and you wish to reach a host called `mumble` in the `bar.edu` domain, you will have to refer to it by the fully-qualified domain name, `mumble.bar.edu`, instead of just `mumble`.

Traditionally, this was allowed by BSD BIND resolvers. However the current version of **BIND** that ships with FreeBSD no longer provides default abbreviations for non-fully qualified domain names other than the domain you are in. So an unqualified host `mumble` must either be found as `mumble.foo.bar.edu`, or it will be searched for in the root domain.

This is different from the previous behavior, where the search continued across `mumble.bar.edu`, and `mumble.edu`. Have a look at RFC 1535 for why this was considered bad practice, or even a security hole.

As a good workaround, you can place the line:

```
search foo.bar.edu bar.edu
```

instead of the previous:

```
domain foo.bar.edu
```

into your `/etc/resolv.conf`. However, make sure that the search order does not go beyond the „boundary between local and public administration”, as RFC 1535 calls it.

2. **sendmail** says mail loops back to myself

This is answered in the **sendmail** FAQ as follows:

I'm getting these error messages:

```
553 MX list for domain.net points back to relay.domain.net
554 <user@domain.net>... Local configuration error
```

How can I solve this problem?

You have asked mail to the domain (e.g., `domain.net`) to be forwarded to a specific host (in this case, `relay.domain.net`) by using an MX record, but the relay machine does not recognize itself as `domain.net`. Add `domain.net` to `/etc/mail/local-host-names` [known as `/etc/sendmail.cw` prior to version 8.10] (if you are using `FEATURE(use_cw_file)`) or add „Cw domain.net”

to /etc/mail/sendmail.cf.

The **sendmail** FAQ can be found at <http://www.sendmail.org/faq/> and is recommended reading if you want to do any „tweaking” of your mail setup.

3. How can I run a mail server on a dial-up PPP host?

You want to connect a FreeBSD box on a LAN to the Internet. The FreeBSD box will be a mail gateway for the LAN. The PPP connection is non-dedicated.

There are at least two ways to do this. One way is to use UUCP.

Another way is to get a full-time Internet server to provide secondary MX services for your domain. For example, if your company’s domain is `example.com` and your Internet service provider has set `example.net` up to provide secondary MX services to your domain:

<code>example.com.</code>	MX	10	<code>example.com.</code>
	MX	20	<code>example.net.</code>

Only one host should be specified as the final recipient (add `Cw example.com` in `/etc/mail/sendmail.cf` on `example.com`).

When the sending `sendmail` is trying to deliver the mail it will try to connect to you (`example.com`) over the modem link. It will most likely time out because you are not online. The program **sendmail** will automatically deliver it to the secondary MX site, i.e. your Internet provider (`example.net`). The secondary MX site will then periodically try to connect to your host and deliver the mail to the primary MX host (`example.com`).

You might want to use something like this as a login script:

```
#!/bin/sh
# Put me in /usr/local/bin/pppmyisp
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppmyisp
```

If you are going to create a separate login script for a user you could use `sendmail -qRexample.com` instead in the script above. This will force all mail in your queue for `example.com` to be processed immediately.

A further refinement of the situation is as follows:

Message stolen from the FreeBSD Internet service provider’s mailing list
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp>).

```
> we provide the secondary MX for a customer. The customer connects to
> our services several times a day automatically to get the mails to
> his primary MX (We do not call his site when a mail for his domains
> arrived). Our sendmail sends the mailqueue every 30 minutes. At the
> moment he has to stay 30 minutes online to be sure that all mail is
> gone to the primary MX.
>
> Is there a command that would initiate sendmail to send all the mails
> now? The user has not root-privileges on our machine of course.
```

In the „privacy flags” section of `sendmail.cf`, there is a definition `Opgoaway,restrictqrun`

Remove restrictqrun to allow non-root users to start the queue processing. You might also like to rearrange the MXs. We are the 1st MX for our customers like this, and we have defined:

```
# If we are the best MX for a host, try directly instead of generating
# local config error.
OwTrue
```

That way a remote site will deliver straight to you, without trying the customer connection. You then send to your customer. Only works for „hosts“, so you need to get your customer to name their mail machine „customer.com“ as well as „hostname.customer.com“ in the DNS. Just put an A record in the DNS for „customer.com“.

4. Why do I keep getting Relaying Denied errors when sending mail from other hosts?

In default FreeBSD installations, **sendmail** is configured to only send mail from the host it is running on. For example, if a POP server is available, then users will be able to check mail from school, work, or other remote locations but they still will not be able to send outgoing emails from outside locations. Typically, a few moments after the attempt, an email will be sent from **MAILER-DAEMON** with a 5.7 Relaying Denied error message.

There are several ways to get around this. The most straightforward solution is to put your ISP's address in a relay-domains file at /etc/mail/relay-domains. A quick way to do this would be:

```
# echo "your.isp.example.com" > /etc/mail/relay-domains
```

After creating or editing this file you must restart **sendmail**. This works great if you are a server administrator and do not wish to send mail locally, or would like to use a point and click client/system on another machine or even another ISP. It is also very useful if you only have one or two email accounts set up. If there is a large number of addresses to add, you can simply open this file in your favorite text editor and then add the domains, one per line:

```
your.isp.example.com
other.isp.example.net
users-isp.example.org
www.example.org
```

Now any mail sent through your system, by any host in this list (provided the user has an account on your system), will succeed. This is a very nice way to allow users to send mail from your system remotely without allowing people to send SPAM through your system.

24.6. Advanced Topics

The following section covers more involved topics such as mail configuration and setting up mail for your entire domain.

24.6.1. Basic Configuration

Out of the box, you should be able to send email to external hosts as long as you have set up `/etc/resolv.conf` or are running your own name server. If you would like to have mail for your host delivered to the MTA (e.g., **sendmail**) on your own FreeBSD host, there are two methods:

- Run your own name server and have your own domain. For example, `FreeBSD.org`
- Get mail delivered directly to your host. This is done by delivering mail directly to the current DNS name for your machine. For example, `example.FreeBSD.org`.

Regardless of which of the above you choose, in order to have mail delivered directly to your host, it must have a permanent static IP address (not a dynamic address, as with most PPP dial-up configurations). If you are behind a firewall, it must pass SMTP traffic on to you. If you want to receive mail directly at your host, you need to be sure of either of two things:

- Make sure that the (lowest-numbered) MX record in your DNS points to your host's IP address.
- Make sure there is no MX entry in your DNS for your host.

Either of the above will allow you to receive mail directly at your host.

Try this:

```
# hostname
example.FreeBSD.org
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
```

If that is what you see, mail directly to `<yourlogin@example.FreeBSD.org>` should work without problems (assuming **sendmail** is running correctly on `example.FreeBSD.org`).

If instead you see something like this:

```
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
example.FreeBSD.org mail is handled (pri=10) by hub.FreeBSD.org
```

All mail sent to your host (`example.FreeBSD.org`) will end up being collected on hub under the same username instead of being sent directly to your host.

The above information is handled by your DNS server. The DNS record that carries mail routing information is the *Mail eXchange* entry. If no MX record exists, mail will be delivered directly to the host by way of its IP address.

The MX entry for `freefall.FreeBSD.org` at one time looked like this:

freefall	MX	30	mail.crl.net
freefall	MX	40	agora.rdrop.com
freefall	MX	10	freefall.FreeBSD.org
freefall	MX	20	who.cdrom.com

As you can see, `freefall` had many MX entries. The lowest MX number is the host that receives mail directly if available; if it is not accessible for some reason, the others (sometimes called „backup MXes”) accept messages temporarily, and pass it along when a lower-numbered host becomes available, eventually to the lowest-numbered host.

Alternate MX sites should have separate Internet connections from your own in order to be most useful. Your ISP or another friendly site should have no problem providing this service for you.

24.6.2. Mail for Your Domain

In order to set up a „mailhost” (a.k.a. mail server) you need to have any mail sent to various workstations directed to it. Basically, you want to „claim” any mail for any hostname in your domain (in this case *.FreeBSD.org) and divert it to your mail server so your users can receive their mail on the master mail server.

To make life easiest, a user account with the same *username* should exist on both machines. Use `adduser(8)` to do this.

The mailhost you will be using must be the designated mail exchanger for each workstation on the network. This is done in your DNS configuration like so:

```
example.FreeBSD.org      A          204.216.27.XX          ; Workstation
                        MX          10  hub.FreeBSD.org        ; Mailhost
```

This will redirect mail for the workstation to the mailhost no matter where the A record points. The mail is sent to the MX host.

You cannot do this yourself unless you are running a DNS server. If you are not, or cannot run your own DNS server, talk to your ISP or whoever provides your DNS.

If you are doing virtual email hosting, the following information will come in handy. For this example, we will assume you have a customer with his own domain, in this case `customer1.org`, and you want all the mail for `customer1.org` sent to your mailhost, `mail.myhost.com`. The entry in your DNS should look like this:

```
customer1.org            MX          10          mail.myhost.com
```

You do *not* need an A record for `customer1.org` if you only want to handle email for that domain.

Notatka: Be aware that pinging `customer1.org` will not work unless an A record exists for it.

The last thing that you must do is tell **sendmail** on your mailhost what domains and/or hostnames it should be accepting mail for. There are a few different ways this can be done. Either of the following will work:

- Add the hosts to your `/etc/mail/local-host-names` file if you are using the `FEATURE(use_cw_file)`. If you are using a version of **sendmail** earlier than 8.10, the file is `/etc/sendmail.cw`.
- Add a `Cyour.host.com` line to your `/etc/sendmail.cf` or `/etc/mail/sendmail.cf` if you are using **sendmail** 8.10 or higher.

24.7. SMTP with UUCP

The **sendmail** configuration that ships with FreeBSD is designed for sites that connect directly to the Internet. Sites that wish to exchange their mail via UUCP must install another **sendmail** configuration file.

Tweaking `/etc/mail/sendmail.cf` manually is an advanced topic. **sendmail** version 8 generates config files via m4(1) preprocessing, where the actual configuration occurs on a higher abstraction level. The m4(1) configuration files can be found under `/usr/src/usr.sbin/sendmail/cf`.

If you did not install your system with full sources, the **sendmail** configuration set has been broken out into a separate source distribution tarball. Assuming you have your FreeBSD source code CDROM mounted, do:

```
# cd /cdrom/src
# cat scontrib.?? | tar xzf - -C /usr/src/contrib/sendmail
```

This extracts to only a few hundred kilobytes. The file `README` in the `cf` directory can serve as a basic introduction to m4(1) configuration.

The best way to support UUCP delivery is to use the `mailertable` feature. This creates a database that **sendmail** can use to make routing decisions.

First, you have to create your `.mc` file. The directory `/usr/src/usr.sbin/sendmail/cf/cf` contains a few examples. Assuming you have named your file `foo.mc`, all you need to do in order to convert it into a valid `sendmail.cf` is:

```
# cd /usr/src/usr.sbin/sendmail/cf/cf
# make foo.cf
# cp foo.cf /etc/mail/sendmail.cf
```

A typical `.mc` file might look like:

```
VERSIONID('Your version number') OSTYPE(bsd4.4)

FEATURE(accept_unresolvable_domains)
FEATURE(nocanonify)
FEATURE(mailertable, 'hash -o /etc/mail/mailertable')

define('UUCP_RELAY', your.uucp.relay)
define('UUCP_MAX_SIZE', 200000)
define('confDONT_PROBE_INTERFACES')

MAILER(local)
MAILER(smtp)
MAILER(uucp)

Cw    your.alias.host.name
Cw    youruucpnodename.UUCP
```

The lines containing `accept_unresolvable_domains`, `nocanonify`, and `confDONT_PROBE_INTERFACES` features will prevent any usage of the DNS during mail delivery. The `UUCP_RELAY` clause is needed to support UUCP delivery. Simply put an Internet hostname there that is able to handle `.UUCP` pseudo-domain addresses; most likely, you will enter the mail relay of your ISP there.

Once you have this, you need an `/etc/mail/mailertable` file. If you have only one link to the outside that is used for all your mails, the following file will suffice:

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
.
    uucp-dom:your.uucp.relay
```

A more complex example might look like this:

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
#
horus.interface-business.de    uucp-dom:horus
.interface-business.de        uucp-dom:if-bus
interface-business.de          uucp-dom:if-bus
.heep.sax.de                   smtp8:%1
horus.UUCP                     uucp-dom:horus
if-bus.UUCP                    uucp-dom:if-bus
.                               uucp-dom:
```

The first three lines handle special cases where domain-addressed mail should not be sent out to the default route, but instead to some UUCP neighbor in order to „shortcut” the delivery path. The next line handles mail to the local Ethernet domain that can be delivered using SMTP. Finally, the UUCP neighbors are mentioned in the .UUCP pseudo-domain notation, to allow for a *uucp-neighbor !recipient* override of the default rules. The last line is always a single dot, matching everything else, with UUCP delivery to a UUCP neighbor that serves as your universal mail gateway to the world. All of the node names behind the *uucp-dom:* keyword must be valid UUCP neighbors, as you can verify using the command `uname`.

As a reminder that this file needs to be converted into a DBM database file before use. The command line to accomplish this is best placed as a comment at the top of the `mailertable` file. You always have to execute this command each time you change your `mailertable` file.

Final hint: if you are uncertain whether some particular mail routing would work, remember the `-bt` option to **sendmail**. It starts **sendmail** in *address test mode*; simply enter `3,0`, followed by the address you wish to test for the mail routing. The last line tells you the used internal mail agent, the destination host this agent will be called with, and the (possibly translated) address. Leave this mode by typing **Ctrl+D**.

```
% sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 foo@example.com
canonify          input: foo @ example . com
...
parse             returns: $# uucp-dom $@ your.uucp.relay $: foo < @ example . com . >
> ^D
```

24.8. Setting Up to Send Only

Contributed by Bill Moran.

There are many instances where you may only want to send mail through a relay. Some examples are:

- Your computer is a desktop machine, but you want to use programs such as `send-pr(1)`. To do so, you should use your ISP's mail relay.
- The computer is a server that does not handle mail locally, but needs to pass off all mail to a relay for processing.

Just about any MTA is capable of filling this particular niche. Unfortunately, it can be very difficult to properly configure a full-featured MTA just to handle offloading mail. Programs such as **sendmail** and **postfix** are largely overkill for this use.

Additionally, if you are using a typical Internet access service, your agreement may forbid you from running a „mail server”.

The easiest way to fulfill those needs is to install the `mail/ssmtp` port. Execute the following commands as `root`:

```
# cd /usr/ports/mail/ssmtp
# make install replace clean
```

Once installed, `mail/ssmtp` can be configured with a four-line file located at `/usr/local/etc/ssmtp/ssmtp.conf`:

```
root=yourrealemail@example.com
mailhub=mail.example.com
rewriteDomain=example.com
hostname=_HOSTNAME_
```

Make sure you use your real email address for `root`. Enter your ISP’s outgoing mail relay in place of `mail.example.com` (some ISPs call this the „outgoing mail server” or „SMTP server”).

Make sure you disable **sendmail**, including the outgoing mail service. See Sekcja 24.4.2 for details.

`mail/ssmtp` has some other options available. See the example configuration file in `/usr/local/etc/ssmtp` or the manual page of **ssmtp** for some examples and more information.

Setting up **ssmtp** in this manner will allow any software on your computer that needs to send mail to function properly, while not violating your ISP’s usage policy or allowing your computer to be hijacked for spamming.

24.9. Using Mail with a Dialup Connection

If you have a static IP address, you should not need to adjust anything from the defaults. Set your host name to your assigned Internet name and **sendmail** will do the rest.

If you have a dynamically assigned IP number and use a dialup PPP connection to the Internet, you will probably have a mailbox on your ISP’s mail server. Let’s assume your ISP’s domain is `example.net`, and that your user name is `user`, you have called your machine `bsd.home`, and your ISP has told you that you may use `relay.example.net` as a mail relay.

In order to retrieve mail from your mailbox, you must install a retrieval agent. The **fetchmail** utility is a good choice as it supports many different protocols. This program is available as a package or from the Ports Collection (`mail/fetchmail`). Usually, your ISP will provide POP. If you are using user PPP, you can automatically fetch your mail when an Internet connection is established with the following entry in `/etc/ppp/ppp.linkup`:

```
MYADDR:
!bg su user -c fetchmail
```

If you are using **sendmail** (as shown below) to deliver mail to non-local accounts, you probably want to have **sendmail** process your mailqueue as soon as your Internet connection is established. To do this, put this command after the `fetchmail` command in `/etc/ppp/ppp.linkup`:

```
!bg su user -c "sendmail -q"
```

Assume that you have an account for user on `bsd.home`. In the home directory of user on `bsd.home`, create a `.fetchmailrc` file:

```
poll example.net protocol pop3 fetchall pass MySecret
```

This file should not be readable by anyone except user as it contains the password `MySecret`.

In order to send mail with the correct `from:` header, you must tell **sendmail** to use `<user@example.net>` rather than `<user@bsd.home>`. You may also wish to tell **sendmail** to send all mail via `relay.example.net`, allowing quicker mail transmission.

The following `.mc` file should suffice:

```
VERSIONID('bsd.home.mc version 1.0')
OSTYPE(bsd4.4)dnl
FEATURE(nouucp)dnl
MAILER(local)dnl
MAILER(smtp)dnl
Cwlocalhost
Cwbsd.home
MASQUERADE_AS('example.net')dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(nocanonify)dnl
FEATURE(nodns)dnl
define('SMART_HOST', 'relay.example.net')
Dmbsd.home
define('confDOMAIN_NAME', 'bsd.home')dnl
define('confDELIVERY_MODE', 'deferred')dnl
```

Refer to the previous section for details of how to turn this `.mc` file into a `sendmail.cf` file. Also, do not forget to restart **sendmail** after updating `sendmail.cf`.

24.10. SMTP Authentication

Written by James Gorham.

Having SMTP Authentication in place on your mail server has a number of benefits. SMTP Authentication can add another layer of security to **sendmail**, and has the benefit of giving mobile users who switch hosts the ability to use the same mail server without the need to reconfigure their mail client settings each time.

1. Install `security/cyrus-sasl` from the ports. You can find this port in `security/cyrus-sasl`. `security/cyrus-sasl` has a number of compile time options to choose from and, for the method we will be using here, make sure to select the `pwcheck` option.
2. After installing `security/cyrus-sasl`, edit `/usr/local/lib/sasl/Sendmail.conf` (or create it if it does not exist) and add the following line:

```
pwcheck_method: passwd
```

This method will enable **sendmail** to authenticate against your FreeBSD `passwd` database. This saves the trouble of creating a new set of usernames and passwords for each user that needs to use SMTP authentication, and keeps the login and mail password the same.

3. Now edit `/etc/make.conf` and add the following lines:

```
SENDMAIL_CFLAGS=-I/usr/local/include/sasl1 -DSASL
SENDMAIL_LDFLAGS=-L/usr/local/lib
SENDMAIL_LDADD=-lsasl
```

These lines will give **sendmail** the proper configuration options for linking to `cyrus-sasl` at compile time. Make sure that `cyrus-sasl` has been installed before recompiling **sendmail**.

4. Recompile **sendmail** by executing the following commands:

```
# cd /usr/src/usr.sbin/sendmail
# make cleandir
# make obj
# make
# make install
```

The compile of **sendmail** should not have any problems if `/usr/src` has not been changed extensively and the shared libraries it needs are available.

5. After **sendmail** has been compiled and reinstalled, edit your `/etc/mail/freebsd.mc` file (or whichever file you use as your `.mc` file. Many administrators choose to use the output from `hostname(1)` as the `.mc` file for uniqueness). Add these lines to it:

```
dnl set SASL options
TRUST_AUTH_MECH('GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
define('confAUTH_MECHANISMS', 'GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
define('confDEF_AUTH_INFO', '/etc/mail/auth-info')dnl
```

These options configure the different methods available to **sendmail** for authenticating users. If you would like to use a method other than **pwcheck**, please see the included documentation.

6. Finally, run `make(1)` while in `/etc/mail`. That will run your new `.mc` file and create a `.cf` file named `freebsd.cf` (or whatever name you have used for your `.mc` file). Then use the command `make install restart`, which will copy the file to `sendmail.cf`, and will properly restart **sendmail**. For more information about this process, you should refer to `/etc/mail/Makefile`.

If all has gone correctly, you should be able to enter your login information into the mail client and send a test message. For further investigation, set the `LogLevel` of **sendmail** to 13 and watch `/var/log/maillog` for any errors.

You may wish to add the following line to `/etc/rc.conf` so this service will be available after every system boot:

```
cyrus_pwcheck_enable="YES"
```

This will ensure the initialization of `SMTP_AUTH` upon system boot.

For more information, please see the **sendmail** page regarding SMTP authentication (<http://www.sendmail.org/~ca/email/auth.html>).

24.11. Mail User Agents

Contributed by Marc Silver.

A Mail User Agent (MUA) is an application that is used to send and receive email. Furthermore, as email „evolves” and becomes more complex, MUA’s are becoming increasingly powerful in the way they interact with email; this

gives users increased functionality and flexibility. FreeBSD contains support for numerous mail user agents, all of which can be easily installed using the FreeBSD Ports Collection. Users may choose between graphical email clients such as **evolution** or **balsa**, console based clients such as **mutt**, **pine** or **mail**, or the web interfaces used by some large organizations.

24.11.1. mail

mail(1) is the default Mail User Agent (MUA) in FreeBSD. It is a console based MUA that offers all the basic functionality required to send and receive text-based email, though it is limited in interaction abilities with attachments and can only support local mailboxes.

Although **mail** does not natively support interaction with POP or IMAP servers, these mailboxes may be downloaded to a local **mbox** file using an application such as **fetchmail**, which will be discussed later in this chapter (Sekcja 24.12).

In order to send and receive email, simply invoke the **mail** command as per the following example:

```
% mail
```

The contents of the user mailbox in **/var/mail** are automatically read by the **mail** utility. Should the mailbox be empty, the utility exits with a message indicating that no mails could be found. Once the mailbox has been read, the application interface is started, and a list of messages will be displayed. Messages are automatically numbered, as can be seen in the following example:

```
Mail version 8.1 6/6/93.  Type ? for help.
"/var/mail/marcs": 3 messages 3 new
>N  1 root@localhost      Mon Mar  8 14:05  14/510  "test "
   N  2 root@localhost      Mon Mar  8 14:05  14/509  "user account"
   N  3 root@localhost      Mon Mar  8 14:05  14/509  "sample"
```

Messages can now be read by using the **t** **mail** command, suffixed by the message number that should be displayed. In this example, we will read the first email:

```
& t 1
Message 1:
From root@localhost  Mon Mar  8 14:05:52 2004
X-Original-To: marcs@localhost
Delivered-To: marcs@localhost
To: marcs@localhost
Subject: test
Date: Mon,  8 Mar 2004 14:05:52 +0200 (SAST)
From: root@localhost (Charlie Root)
```

This is a test message, please reply if you receive it.

As can be seen in the example above, the **t** key will cause the message to be displayed with full headers. To display the list of messages again, the **h** key should be used.

If the email requires a response, you may use **mail** to reply, by using either the **R** or **r** **mail** keys. The **R** key instructs **mail** to reply only to the sender of the email, while **r** replies not only to the sender, but also to other recipients of the message. You may also suffix these commands with the mail number which you would like make a reply to. Once this has been done, the response should be entered, and the end of the message should be marked by a single **.** on a new line. An example can be seen below:

```
& R 1
To: root@localhost
Subject: Re: test
```

Thank you, I did get your email.

.

EOT

In order to send new email, the **m** key should be used, followed by the recipient email address. Multiple recipients may also be specified by separating each address with the , delimiter. The subject of the message may then be entered, followed by the message contents. The end of the message should be specified by putting a single . on a new line.

```
& mail root@localhost
Subject: I mastered mail
```

Now I can send and receive email using mail ... :)

.

EOT

While inside the `mail` utility, the `?` command may be used to display help at any time, the `mail(1)` manual page should also be consulted for more help with `mail`.

Notatka: As previously mentioned, the `mail(1)` command was not originally designed to handle attachments, and thus deals with them very poorly. Newer MUAs such as **mutt** handle attachments in a much more intelligent way. But should you still wish to use the `mail` command, the `converters/mpack` port may be of considerable use.

24.11.2. mutt

mutt is a small yet very powerful Mail User Agent, with excellent features, just some of which include:

- The ability to thread messages;
- PGP support for digital signing and encryption of email;
- MIME Support;
- Maildir Support;
- Highly customizable.

All of these features help to make **mutt** one of the most advanced mail user agents available. See <http://www.mutt.org> for more information on **mutt**.

The stable version of **mutt** may be installed using the `mail/mutt` port, while the current development version may be installed via the `mail/mutt-devel` port. After the port has been installed, **mutt** can be started by issuing the following command:

```
% mutt
```

mutt will automatically read the contents of the user mailbox in `/var/mail` and display the contents if applicable. If no mails are found in the user mailbox, then **mutt** will wait for commands from the user. The example below shows **mutt** displaying a list of messages:

```
q:Quit  d:Del  u:Undel  s:Save  m:Mail  r:Reply  g:Group  ?:Help
1 N    Mar 09 Super-User      ( 1) test
2 N    Mar 09 Super-User      ( 1) user account
3 N    Mar 09 Super-User      ( 1) sample

--*Mutt: /var/mail/marcs [Msgs:3 New:3 1.6K]---(date/date)----- (all)---
```

In order to read an email, simply select it using the cursor keys, and press the **Enter** key. An example of **mutt** displaying email can be seen below:

```
i:Exit  -:PrevPg <Space>:NextPg u:View Attachm. d:Del  r:Reply  j:Next ?:Help
X-Original-To: marcs@localhost
Delivered-To: marcs@localhost
To: marcs@localhost
Subject: test
Date: Tue, 9 Mar 2004 10:28:36 +0200 (SAST)
From: Super-User <root@localhost>

This is a test message, please reply if you receive it.

--N - 1/1: Super-User      test      -- (all)
```

As with the `mail(1)` command, **mutt** allows users to reply only to the sender of the message as well as to all recipients. To reply only to the sender of the email, use the **r** keyboard shortcut. To send a group reply, which will be sent to the original sender as well as all the message recipients, use the **g** shortcut.

Notatka: **mutt** makes use of the `vi(1)` command as an editor for creating and replying to emails. This may be customized by the user by creating or editing their own `.muttrc` file in their home directory and setting the `editor` variable.

In order to compose a new mail message, press **m**. After a valid subject has been given, **mutt** will start `vi(1)` and the mail can be written. Once the contents of the mail are complete, save and quit from `vi` and **mutt** will resume,

displaying a summary screen of the mail that is to be delivered. In order to send the mail, press **y**. An example of the summary screen can be seen below:

```

y:Send q:Abort t:To c:CC s:Subj a:Attach file d:Descrip ?:Help
  From: Marc Silver <marcs@localhost>
  To: Super-User <root@localhost>
  Cc:
  Bcc:
  Subject: Re: test
  Reply-To:
  Fcc:
  Security: Clear

-- Attachments
- I 1 /tmp/mutt-bsd-c0hobscQ [text/plain, 7bit, us-ascii, 1.1K]

-- Mutt: Compose [Approx. msg size: 1.1K Atts: 1]-----

```

mutt also contains extensive help, which can be accessed from most of the menus by pressing the **?** key. The top line also displays the keyboard shortcuts where appropriate.

24.11.3. pine

pine is aimed at a beginner user, but also includes some advanced features.

OstrzeżenieThe **pine** software has had several remote vulnerabilities discovered in the past, which allowed remote attackers to execute arbitrary code as users on the local system, by the action of sending a specially-prepared email. All such *known* problems have been fixed, but the **pine** code is written in a very insecure style and the FreeBSD Security Officer believes there are likely to be other undiscovered vulnerabilities. You install **pine** at your own risk.

The current version of **pine** may be installed using the `mail/pine4` port. Once the port has installed, **pine** can be started by issuing the following command:

```
% pine
```

The first time that **pine** is run it displays a greeting page with a brief introduction, as well as a request from the **pine** development team to send an anonymous email message allowing them to judge how many users are using their client. To send this anonymous message, press **Enter**, or alternatively press **E** to exit the greeting without sending an anonymous message. An example of the greeting page can be seen below:

```

PINE 4.58  GREETING TEXT                                     No Messages

<<<This message will appear only once>>>

Welcome to Pine ... a Program for Internet News and Email

We hope you will explore Pine's many capabilities. From the Main Menu,
select Setup/Config to see many of the options available to you. Also
note that all screens have context-sensitive help text available.

SPECIAL REQUEST: This software is made available world-wide as a public
service of the University of Washington in Seattle. In order to justify
continuing development, it is helpful to have an idea of how many people
are using Pine. Are you willing to be counted as a Pine user? Pressing
Return will send an anonymous (meaning, your real email address will not
be revealed) message to the Pine development team at the University of
Washington for purposes of tallying.

Pine is a trademark of the University of Washington.

[ALL of greeting text]
? Help      [E] Exit this greeting      [P] PrevPage  [Z] Print
[Ret] [Be Counted!]                  [SpC] NextPage

```

Users are then presented with the main menu, which can be easily navigated using the cursor keys. This main menu provides shortcuts for the composing new mails, browsing of mail directories, and even the administration of address book entries. Below the main menu, relevant keyboard shortcuts to perform functions specific to the task at hand are shown.

The default directory opened by **pine** is the **inbox**. To view the message index, press **I**, or select the **MESSAGE INDEX** option as seen below:

```

PINE 4.58  MAIN MENU                                         Folder: INBOX  3 Messages

?  HELP          - Get help using Pine
C  COMPOSE MESSAGE - Compose and send a message
I  MESSAGE INDEX  - View messages in current folder
L  FOLDER LIST    - Select a folder to view
A  ADDRESS BOOK   - Update address book
S  SETUP          - Configure Pine Options
Q  QUIT           - Leave the Pine program

Copyright 1989-2003.  PINE is a trademark of the University of Washington.

? Help      [P] PrevCmd      [R] RelNotes
[O] OTHER CMDS [Z] [Index]  [N] NextCmd      [X] KBLock

```

The message index shows messages in the current directory, and can be navigated by using the cursor keys. Highlighted messages can be read by pressing the **Enter** key.

```

PINE 4.58  MESSAGE INDEX                               Folder: INBOX  Message 1 of 3 ANS
-----
A  1 Mar  9 Super-User                                (471) test
A  2 Mar  9 Super-User                                (479) user account
A  3 Mar  9 Super-User                                (473) sample

? Help  < FldrList  P PrevMsg  - PrevPage  D Delete  R Reply
0 OTHER CMDS > [ViewMsg] N NextMsg  Spc NextPage U Undelete F Forward

```

In the screenshot below, a sample message is displayed by **pine**. Keyboard shortcuts are displayed as a reference at the bottom of the screen. An example of one of these shortcuts is the **r** key, which tells the MUA to reply to the current message being displayed.

```

PINE 4.58  MESSAGE TEXT                               Folder: INBOX  Message 1 of 3 ALL ANS
-----
Date: Tue,  9 Mar 2004 10:28:36 +0200 (SAST)
From: Super-User <root@localhost>
To: marcs@localhost
Subject: test

This is a test message, please reply if you receive it.

[ALL of message]
? Help  < MsgIndex  P PrevMsg  - PrevPage  D Delete  R Reply
0 OTHER CMDS > ViewAtch N NextMsg  Spc NextPage U Undelete F Forward

```

Replying to an email in **pine** is done using the **pico** editor, which is installed by default with **pine**. The **pico** utility makes it easy to navigate around the message and is slightly more forgiving on novice users than **vi**(1) or **mail**(1). Once the reply is complete, the message can be sent by pressing **Ctrl+X**. The **pine** application will ask for confirmation.



The **pine** application can be customized using the **SETUP** option from the main menu. Consult <http://www.washington.edu/pine/> for more information.

24.12. Using fetchmail

Contributed by Marc Silver.

fetchmail is a full-featured IMAP and POP client which allows users to automatically download mail from remote IMAP and POP servers and save it into local mailboxes; there it can be accessed more easily. **fetchmail** can be installed using the `mail/fetchmail` port, and offers various features, some of which include:

- Support of POP3, APOP, KPOP, IMAP, ETRN and ODMR protocols.
- Ability to forward mail using SMTP, which allows filtering, forwarding, and aliasing to function normally.
- May be run in daemon mode to check periodically for new messages.
- Can retrieve multiple mailboxes and forward them based on configuration, to different local users.

While it is outside the scope of this document to explain all of **fetchmail**'s features, some basic features will be explained. The **fetchmail** utility requires a configuration file known as `.fetchmailrc`, in order to run correctly. This file includes server information as well as login credentials. Due to the sensitive nature of the contents of this file, it is advisable to make it readable only by the owner, with the following command:

```
% chmod 600 .fetchmailrc
```

The following `.fetchmailrc` serves as an example for downloading a single user mailbox using POP. It tells **fetchmail** to connect to `example.com` using a username of `joesoap` and a password of `XXX`. This example assumes that the user `joesoap` is also a user on the local system.

```
poll example.com protocol pop3 username "joesoap" password "XXX"
```

The next example connects to multiple POP and IMAP servers and redirects to different local usernames where applicable:

```
poll example.com proto pop3:
user "joesoap", with password "XXX", is "jsoap" here;
user "andrea", with password "XXXX";
poll example2.net proto imap:
user "john", with password "XXXXX", is "myth" here;
```

The **fetchmail** utility can be run in daemon mode by running it with the `-d` flag, followed by the interval (in seconds) that **fetchmail** should poll servers listed in the `.fetchmailrc` file. The following example would cause **fetchmail** to poll every 600 seconds:

```
% fetchmail -d 600
```

More information on **fetchmail** can be found at <http://fetchmail.berlios.de/>.

24.13. Using procmail

Contributed by Marc Silver.

The **procmail** utility is an incredibly powerful application used to filter incoming mail. It allows users to define „rules” which can be matched to incoming mails to perform specific functions or to reroute mail to alternative mailboxes and/or email addresses. **procmail** can be installed using the `mail/procmail` port. Once installed, it can be directly integrated into most MTAs; consult your MTA documentation for more information. Alternatively, **procmail** can be integrated by adding the following line to a `.forward` in the home directory of the user utilizing **procmail** features:

```
"|exec /usr/local/bin/procmail || exit 75"
```

The following section will display some basic **procmail** rules, as well as brief descriptions on what they do. These rules, and others must be inserted into a `.procmailrc` file, which must reside in the user’s home directory.

The majority of these rules can also be found in the `procmailex(5)` manual page.

Forward all mail from `<user@example.com>` to an external address of `<goodmail@example2.com>`:

```
:0
* ^From.*user@example.com
! goodmail@example2.com
```

Forward all mails shorter than 1000 bytes to an external address of `<goodmail@example2.com>`:

```
:0
* < 1000
! goodmail@example2.com
```

Send all mail sent to `<alternate@example.com>` into a mailbox called `alternate`:

```
:0
* ^TOalternate@example.com
alternate
```

Send all mail with a subject of „Spam” to `/dev/null`:

```
:0
```

```
^Subject:.*Spam
/dev/null
```

A useful recipe that parses incoming FreeBSD.org mailing lists and places each list in its own mailbox:

```
:0
* ^Sender:.owner-freebsd-\[ ^@\]+@FreeBSD.ORG
{
    LISTNAME=${MATCH}
    :0
    * LISTNAME??^\[ ^@\]+
    FreeBSD-${MATCH}
}
```

Rozdział 25. Network Servers

Reorganized by Murray Stokely.

25.1. Synopsis

This chapter will cover some of the more frequently used network services on UNIX systems. We will cover how to install, configure, test, and maintain many different types of network services. Example configuration files are included throughout this chapter for you to benefit from.

After reading this chapter, you will know:

- How to manage the **inetd** daemon.
- How to set up a network file system.
- How to set up a network information server for sharing user accounts.
- How to set up automatic network settings using DHCP.
- How to set up a domain name server.
- How to set up the **Apache** HTTP Server.
- How to set up a File Transfer Protocol (FTP) Server.
- How to set up a file and print server for Windows clients using **Samba**.
- How to synchronize the time and date, and set up a time server, with the NTP protocol.

Before reading this chapter, you should:

- Understand the basics of the `/etc/rc` scripts.
- Be familiar with basic network terminology.
- Know how to install additional third-party software (Rozdział 4).

25.2. The **inetd** „Super-Server”

Contributed by Chern Lee. Updated for FreeBSD 6.1-RELEASE by The FreeBSD Documentation Project.

25.2.1. Overview

inetd(8) is sometimes referred to as the „Internet Super-Server” because it manages connections for several services. When a connection is received by **inetd**, it determines which program the connection is destined for, spawns the particular process and delegates the socket to it (the program is invoked with the service socket as its standard input, output and error descriptors). Running **inetd** for servers that are not heavily used can reduce the overall system load, when compared to running each daemon individually in stand-alone mode.

Primarily, **inetd** is used to spawn other daemons, but several trivial protocols are handled directly, such as **chargen**, **auth**, and **daytime**.

This section will cover the basics in configuring **inetd** through its command-line options and its configuration file, `/etc/inetd.conf`.

25.2.2. Settings

inetd is initialized through the `rc(8)` system. The `inetd_enable` option is set to `NO` by default, but may be turned on by **sysinstall** during installation, depending on the configuration chosen by the user. Placing:

```
inetd_enable="YES"
```

or

```
inetd_enable="NO"
```

into `/etc/rc.conf` will enable or disable **inetd** starting at boot time. The command:

```
/etc/rc.d/inetd rcvar
```

can be run to display the current effective setting.

Additionally, different command-line options can be passed to **inetd** via the `inetd_flags` option.

25.2.3. Command-Line Options

Like most server daemons, **inetd** has a number of options that it can be passed in order to modify its behaviour. The full list of options reads:

```
inetd [-d] [-l] [-w] [-W] [-c maximum] [-C rate] [-a address | hostname] [-p filename]
[-R rate] [-s maximum] [configuration file]
```

Options can be passed to **inetd** using the `inetd_flags` option in `/etc/rc.conf`. By default, `inetd_flags` is set to `-wW -C 60`, which turns on TCP wrapping for **inetd**'s services, and prevents any single IP address from requesting any service more than 60 times in any given minute.

Novice users may be pleased to note that these parameters usually do not need to be modified, although we mention the rate-limiting options below as they be useful should you find that you are receiving an excessive amount of connections. A full list of options can be found in the `inetd(8)` manual.

-c maximum

Specify the default maximum number of simultaneous invocations of each service; the default is unlimited. May be overridden on a per-service basis with the `max-child` parameter.

-C rate

Specify the default maximum number of times a service can be invoked from a single IP address in one minute; the default is unlimited. May be overridden on a per-service basis with the `max-connections-per-ip-per-minute` parameter.

-R rate

Specify the maximum number of times a service can be invoked in one minute; the default is 256. A rate of 0 allows an unlimited number of invocations.

-s maximum

Specify the maximum number of times a service can be invoked from a single IP address at any one time; the default is unlimited. May be overridden on a per-service basis with the `max-child-per-ip` parameter.

25.2.4. `inetd.conf`

Configuration of **inetd** is done via the file `/etc/inetd.conf`.

When a modification is made to `/etc/inetd.conf`, **inetd** can be forced to re-read its configuration file by running the command:

Przykład 25-1. Reloading the `inetd` configuration file

```
# /etc/rc.d/inetd reload
```

Each line of the configuration file specifies an individual daemon. Comments in the file are preceded by a „#”. The format of each entry in `/etc/inetd.conf` is as follows:

```
service-name
socket-type
protocol
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]
user[:group][[/login-class]]
server-program
server-program-arguments
```

An example entry for the `ftpd(8)` daemon using IPv4 might read:

```
ftp      stream  tcp      nowait  root    /usr/libexec/ftpd      ftpd -l
```

service-name

This is the service name of the particular daemon. It must correspond to a service listed in `/etc/services`. This determines which port **inetd** must listen to. If a new service is being created, it must be placed in `/etc/services` first.

socket-type

Either `stream`, `dgram`, `raw`, or `seqpacket`. `stream` must be used for connection-based, TCP daemons, while `dgram` is used for daemons utilizing the UDP transport protocol.

protocol

One of the following:

Protocol	Explanation
<code>tcp, tcp4</code>	TCP IPv4
<code>udp, udp4</code>	UDP IPv4
<code>tcp6</code>	TCP IPv6
<code>udp6</code>	UDP IPv6

Protocol	Explanation
tcp46	Both TCP IPv4 and v6
udp46	Both UDP IPv4 and v6

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]

`wait|nowait` indicates whether the daemon invoked from **inetd** is able to handle its own socket or not. `dgram` socket types must use the `wait` option, while stream socket daemons, which are usually multi-threaded, should use `nowait`. `wait` usually hands off multiple sockets to a single daemon, while `nowait` spawns a child daemon for each new socket.

The maximum number of child daemons **inetd** may spawn can be set using the `max-child` option. If a limit of ten instances of a particular daemon is needed, a `/10` would be placed after `nowait`. Specifying `/0` allows an unlimited number of children

In addition to `max-child`, two other options which limit the maximum connections from a single place to a particular daemon can be enabled. `max-connections-per-ip-per-minute` limits the number of connections from any particular IP address per minutes, e.g. a value of ten would limit any particular IP address connecting to a particular service to ten attempts per minute. `max-child-per-ip` limits the number of children that can be started on behalf on any single IP address at any moment. These options are useful to prevent intentional or unintentional excessive resource consumption and Denial of Service (DoS) attacks to a machine.

In this field, either of `wait` or `nowait` is mandatory. `max-child`, `max-connections-per-ip-per-minute` and `max-child-per-ip` are optional.

A stream-type multi-threaded daemon without any `max-child`, `max-connections-per-ip-per-minute` or `max-child-per-ip` limits would simply be: `nowait`.

The same daemon with a maximum limit of ten daemons would read: `nowait/10`.

The same setup with a limit of twenty connections per IP address per minute and a maximum total limit of ten child daemons would read: `nowait/10/20`.

These options are utilized by the default settings of the `fingerd(8)` daemon, as seen here:

```
finger stream tcp      nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

Finally, an example of this field with a maximum of 100 children in total, with a maximum of 5 for any one IP address would read: `nowait/100/0/5`.

user

This is the username that the particular daemon should run as. Most commonly, daemons run as the `root` user. For security purposes, it is common to find some servers running as the `daemon` user, or the least privileged `nobody` user.

server-program

The full path of the daemon to be executed when a connection is received. If the daemon is a service provided by **inetd** internally, then `internal` should be used.

server-program-arguments

This works in conjunction with `server-program` by specifying the arguments, starting with `argv[0]`, passed to the daemon on invocation. If `mydaemon -d` is the command line, `mydaemon -d` would be the value of

`server-program-arguments`. Again, if the daemon is an internal service, use `internal` here.

25.2.5. Security

Depending on the choices made at install time, many of **inetd**'s services may be enabled by default. If there is no apparent need for a particular daemon, consider disabling it. Place a „#” in front of the daemon in question in `/etc/inetd.conf`, and then reload the **inetd** configuration. Some daemons, such as **fingerd**, may not be desired at all because they provide information that may be useful to an attacker.

Some daemons are not security-conscious and have long, or non-existent, timeouts for connection attempts. This allows an attacker to slowly send connections to a particular daemon, thus saturating available resources. It may be a good idea to place `max-connections-per-ip-per-minute`, `max-child` or `max-child-per-ip` limitations on certain daemons if you find that you have too many connections.

By default, TCP wrapping is turned on. Consult the `hosts_access(5)` manual page for more information on placing TCP restrictions on various **inetd** invoked daemons.

25.2.6. Miscellaneous

daytime, **time**, **echo**, **discard**, **chargen**, and **auth** are all internally provided services of **inetd**.

The **auth** service provides identity network services, and is configurable to a certain degree, whilst the others are simply on or off.

Consult the `inetd(8)` manual page for more in-depth information.

25.3. Network File System (NFS)

Reorganized and enhanced by Tom Rhodes. Written by Bill Swingle.

Among the many different file systems that FreeBSD supports is the Network File System, also known as NFS. NFS allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files.

Some of the most notable benefits that NFS can provide are:

- Local workstations use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network.
- There is no need for users to have separate home directories on every network machine. Home directories could be set up on the NFS server and made available throughout the network.
- Storage devices such as floppy disks, CDROM drives, and Zip® drives can be used by other machines on the network. This may reduce the number of removable media drives throughout the network.

25.3.1. How NFS Works

NFS consists of at least two main parts: a server and one or more clients. The client remotely accesses the data that is stored on the server machine. In order for this to function properly a few processes have to be configured and running.

The server has to be running the following daemons:

Daemon	Description
nfsd	The NFS daemon which services requests from the NFS clients.
mountd	The NFS mount daemon which carries out the requests that nfsd(8) passes on to it.
rpcbind	This daemon allows NFS clients to discover which port the NFS server is using.

The client can also run a daemon, known as **nfsiod**. The **nfsiod** daemon services the requests from the NFS server. This is optional, and improves performance, but is not required for normal and correct operation. See the nfsiod(8) manual page for more information.

25.3.2. Configuring NFS

NFS configuration is a relatively straightforward process. The processes that need to be running can all start at boot time with a few modifications to your `/etc/rc.conf` file.

On the NFS server, make sure that the following options are configured in the `/etc/rc.conf` file:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

mountd runs automatically whenever the NFS server is enabled.

On the client, make sure this option is present in `/etc/rc.conf`:

```
nfs_client_enable="YES"
```

The `/etc/exports` file specifies which file systems NFS should export (sometimes referred to as „share”). Each line in `/etc/exports` specifies a file system to be exported and which machines have access to that file system. Along with what machines have access to that file system, access options may also be specified. There are many such options that can be used in this file but only a few will be mentioned here. You can easily discover other options by reading over the exports(5) manual page.

Here are a few example `/etc/exports` entries:

The following examples give an idea of how to export file systems, although the settings may be different depending on your environment and network configuration. For instance, to export the `/cdrom` directory to three example machines that have the same domain name as the server (hence the lack of a domain name for each) or have entries in your `/etc/hosts` file. The `-ro` flag makes the exported file system read-only. With this flag, the remote system will not be able to write any changes to the exported file system.

```
/cdrom -ro host1 host2 host3
```

The following line exports `/home` to three hosts by IP address. This is a useful setup if you have a private network without a DNS server configured. Optionally the `/etc/hosts` file could be configured for internal hostnames; please review hosts(5) for more information. The `-allargs` flag allows the subdirectories to be mount points. In other words, it will not mount the subdirectories but permit the client to mount only the directories that are required or needed.

```
/home -allargs 10.0.0.2 10.0.0.3 10.0.0.4
```

The following line exports `/a` so that two clients from different domains may access the file system. The `-maproot=root` flag allows the `root` user on the remote system to write data on the exported file system as `root`. If the `-maproot=root` flag is not specified, then even if a user has `root` access on the remote system, he will not be able to modify files on the exported file system.

```
/a -maproot=root host.example.com box.example.org
```

In order for a client to access an exported file system, the client must have permission to do so. Make sure the client is listed in your `/etc/exports` file.

In `/etc/exports`, each line represents the export information for one file system to one host. A remote host can only be specified once per file system, and may only have one default entry. For example, assume that `/usr` is a single file system. The following `/etc/exports` would be invalid:

```
# Invalid when /usr is one file system
/usr/src client
/usr/ports client
```

One file system, `/usr`, has two lines specifying exports to the same host, `client`. The correct format for this situation is:

```
/usr/src /usr/ports client
```

The properties of one file system exported to a given host must all occur on one line. Lines without a client specified are treated as a single host. This limits how you can export file systems, but for most people this is not an issue.

The following is an example of a valid export list, where `/usr` and `/exports` are local file systems:

```
# Export src and ports to client01 and client02, but only
# client01 has root privileges on it
/usr/src /usr/ports -maproot=root client01
/usr/src /usr/ports client02
# The client machines have root and can mount anywhere
# on /exports. Anyone in the world can mount /exports/obj read-only
/exports -alldirs -maproot=root client01 client02
/exports/obj -ro
```

The **mountd** daemon must be forced to recheck the `/etc/exports` file whenever it has been modified, so the changes can take effect. This can be accomplished either by sending a HUP signal to the running daemon:

```
# kill -HUP `cat /var/run/mountd.pid`
```

or by invoking the `mountd rc(8)` script with the appropriate parameter:

```
# /etc/rc.d/mountd reload
```

Please refer to Sekcja 11.7 for more information about using `rc` scripts.

Alternatively, a reboot will make FreeBSD set everything up properly. A reboot is not necessary though. Executing the following commands as `root` should start everything up.

On the NFS server:

```
# rpcbind
# nfsd -u -t -n 4
```

```
# mountd -r
```

On the NFS client:

```
# nfsiod -n 4
```

Now everything should be ready to actually mount a remote file system. In these examples the server's name will be `server` and the client's name will be `client`. If you only want to temporarily mount a remote file system or would rather test the configuration, just execute a command like this as `root` on the client:

```
# mount server:/home /mnt
```

This will mount the `/home` directory on the server at `/mnt` on the client. If everything is set up correctly you should be able to enter `/mnt` on the client and see all the files that are on the server.

If you want to automatically mount a remote file system each time the computer boots, add the file system to the `/etc/fstab` file. Here is an example:

```
server:/home    /mnt    nfs     rw      0        0
```

The `fstab(5)` manual page lists all the available options.

25.3.3. Practical Uses

NFS has many practical uses. Some of the more common ones are listed below:

- Set several machines to share a CDROM or other media among them. This is cheaper and often a more convenient method to install software on multiple machines.
- On large networks, it might be more convenient to configure a central NFS server in which to store all the user home directories. These home directories can then be exported to the network so that users would always have the same home directory, regardless of which workstation they log in to.
- Several machines could have a common `/usr/ports/distfiles` directory. That way, when you need to install a port on several machines, you can quickly access the source without downloading it on each machine.

25.3.4. Automatic Mounts with amd

Contributed by Wylie Stilwell. Rewritten by Chern Lee.

`amd(8)` (the automatic mounter daemon) automatically mounts a remote file system whenever a file or directory within that file system is accessed. Filesystems that are inactive for a period of time will also be automatically unmounted by **amd**. Using **amd** provides a simple alternative to permanent mounts, as permanent mounts are usually listed in `/etc/fstab`.

amd operates by attaching itself as an NFS server to the `/host` and `/net` directories. When a file is accessed within one of these directories, **amd** looks up the corresponding remote mount and automatically mounts it. `/net` is used to mount an exported file system from an IP address, while `/host` is used to mount an export from a remote hostname.

An access to a file within `/host/foobar/usr` would tell **amd** to attempt to mount the `/usr` export on the host `foobar`.

Przykład 25-2. Mounting an Export with amd

You can view the available mounts of a remote host with the `showmount` command. For example, to view the mounts of a host named `foobar`, you can use:

```
% showmount -e foobar
Exports list on foobar:
/usr                      10.10.10.0
/a                        10.10.10.0
% cd /host/foobar/usr
```

As seen in the example, the `showmount` shows `/usr` as an export. When changing directories to `/host/foobar/usr`, **amd** attempts to resolve the hostname `foobar` and automatically mount the desired export.

amd can be started by the startup scripts by placing the following lines in `/etc/rc.conf`:

```
amd_enable="YES"
```

Additionally, custom flags can be passed to **amd** from the `amd_flags` option. By default, `amd_flags` is set to:

```
amd_flags="-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map"
```

The `/etc/amd.map` file defines the default options that exports are mounted with. The `/etc/amd.conf` file defines some of the more advanced features of **amd**.

Consult the `amd(8)` and `amd.conf(5)` manual pages for more information.

25.3.5. Problems Integrating with Other Systems

Contributed by John Lind.

Certain Ethernet adapters for ISA PC systems have limitations which can lead to serious network problems, particularly with NFS. This difficulty is not specific to FreeBSD, but FreeBSD systems are affected by it.

The problem nearly always occurs when (FreeBSD) PC systems are networked with high-performance workstations, such as those made by Silicon Graphics, Inc., and Sun Microsystems, Inc. The NFS mount will work fine, and some operations may succeed, but suddenly the server will seem to become unresponsive to the client, even though requests to and from other systems continue to be processed. This happens to the client system, whether the client is the FreeBSD system or the workstation. On many systems, there is no way to shut down the client gracefully once this problem has manifested itself. The only solution is often to reset the client, because the NFS situation cannot be resolved.

Though the „correct” solution is to get a higher performance and capacity Ethernet adapter for the FreeBSD system, there is a simple workaround that will allow satisfactory operation. If the FreeBSD system is the *server*, include the option `-w=1024` on the mount from the client. If the FreeBSD system is the *client*, then mount the NFS file system with the option `-r=1024`. These options may be specified using the fourth field of the `fstab` entry on the client for automatic mounts, or by using the `-o` parameter of the `mount(8)` command for manual mounts.

It should be noted that there is a different problem, sometimes mistaken for this one, when the NFS servers and clients are on different networks. If that is the case, make *certain* that your routers are routing the necessary UDP information, or you will not get anywhere, no matter what else you are doing.

In the following examples, `fastws` is the host (interface) name of a high-performance workstation, and `freebox` is the host (interface) name of a FreeBSD system with a lower-performance Ethernet adapter. Also, `/sharedfs` will be

the exported NFS file system (see `exports(5)`), and `/project` will be the mount point on the client for the exported file system. In all cases, note that additional options, such as `hard` or `soft` and `bg` may be desirable in your application.

Examples for the FreeBSD system (`freebox`) as the client in `/etc/fstab` on `freebox`:

```
fastws:/sharedfs /project nfs rw,-r=1024 0 0
```

As a manual mount command on `freebox`:

```
# mount -t nfs -o -r=1024 fastws:/sharedfs /project
```

Examples for the FreeBSD system as the server in `/etc/fstab` on `fastws`:

```
freebox:/sharedfs /project nfs rw,-w=1024 0 0
```

As a manual mount command on `fastws`:

```
# mount -t nfs -o -w=1024 freebox:/sharedfs /project
```

Nearly any 16-bit Ethernet adapter will allow operation without the above restrictions on the read or write size.

For anyone who cares, here is what happens when the failure occurs, which also explains why it is unrecoverable. NFS typically works with a „block” size of 8 K (though it may do fragments of smaller sizes). Since the maximum Ethernet packet is around 1500 bytes, the NFS „block” gets split into multiple Ethernet packets, even though it is still a single unit to the upper-level code, and must be received, assembled, and *acknowledged* as a unit. The high-performance workstations can pump out the packets which comprise the NFS unit one right after the other, just as close together as the standard allows. On the smaller, lower capacity cards, the later packets overrun the earlier packets of the same unit before they can be transferred to the host and the unit as a whole cannot be reconstructed or acknowledged. As a result, the workstation will time out and try again, but it will try again with the entire 8 K unit, and the process will be repeated, ad infinitum.

By keeping the unit size below the Ethernet packet size limitation, we ensure that any complete Ethernet packet received can be acknowledged individually, avoiding the deadlock situation.

Overruns may still occur when a high-performance workstations is slamming data out to a PC system, but with the better cards, such overruns are not guaranteed on NFS „units”. When an overrun occurs, the units affected will be retransmitted, and there will be a fair chance that they will be received, assembled, and acknowledged.

25.4. Network Information System (NIS/YP)

Written by Bill Swingle. Enhanced by Eric Ogren i Udo Erdelhoff.

25.4.1. What Is It?

NIS, which stands for Network Information Services, was developed by Sun Microsystems to centralize administration of UNIX (originally SunOS) systems. It has now essentially become an industry standard; all major UNIX like systems (Solaris, HP-UX, AIX®, Linux, NetBSD, OpenBSD, FreeBSD, etc) support NIS.

NIS was formerly known as Yellow Pages, but because of trademark issues, Sun changed the name. The old term (and `yp`) is still often seen and used.

It is a RPC-based client/server system that allows a group of machines within an NIS domain to share a common set of configuration files. This permits a system administrator to set up NIS client systems with only minimal configuration data and add, remove or modify configuration data from a single location.

It is similar to the Windows NT® domain system; although the internal implementation of the two are not at all similar, the basic functionality can be compared.

25.4.2. Terms/Processes You Should Know

There are several terms and several important user processes that you will come across when attempting to implement NIS on FreeBSD, whether you are trying to create an NIS server or act as an NIS client:

Term	Description
NIS domainname	An NIS master server and all of its clients (including its slave servers) have a NIS domainname. Similar to an Windows NT domain name, the NIS domainname does not have anything to do with DNS.
rpcbind	Must be running in order to enable RPC (Remote Procedure Call, a network protocol used by NIS). If rpcbind is not running, it will be impossible to run an NIS server, or to act as an NIS client.
ypbind	„Binds” an NIS client to its NIS server. It will take the NIS domainname from the system, and using RPC, connect to the server. ypbind is the core of client-server communication in an NIS environment; if ypbind dies on a client machine, it will not be able to access the NIS server.
ypserv	Should only be running on NIS servers; this is the NIS server process itself. If ypserv (8) dies, then the server will no longer be able to respond to NIS requests (hopefully, there is a slave server to take over for it). There are some implementations of NIS (but not the FreeBSD one), that do not try to reconnect to another server if the server it used before dies. Often, the only thing that helps in this case is to restart the server process (or even the whole server) or the ypbind process on the client.
rpc.yppasswdd	Another process that should only be running on NIS master servers; this is a daemon that will allow NIS clients to change their NIS passwords. If this daemon is not running, users will have to login to the NIS master server and change their passwords there.

25.4.3. How Does It Work?

There are three types of hosts in an NIS environment: master servers, slave servers, and clients. Servers act as a central repository for host configuration information. Master servers hold the authoritative copy of this information, while slave servers mirror this information for redundancy. Clients rely on the servers to provide this information to them.

Information in many files can be shared in this manner. The `master.passwd`, `group`, and `hosts` files are commonly shared via NIS. Whenever a process on a client needs information that would normally be found in these files locally, it makes a query to the NIS server that it is bound to instead.

25.4.3.1. Machine Types

- A *NIS master server*. This server, analogous to a Windows NT primary domain controller, maintains the files used by all of the NIS clients. The `passwd`, `group`, and other various files used by the NIS clients live on the master server.

Notatka: It is possible for one machine to be an NIS master server for more than one NIS domain. However, this will not be covered in this introduction, which assumes a relatively small-scale NIS environment.

- *NIS slave servers*. Similar to the Windows NT backup domain controllers, NIS slave servers maintain copies of the NIS master's data files. NIS slave servers provide the redundancy, which is needed in important environments. They also help to balance the load of the master server: NIS Clients always attach to the NIS server whose response they get first, and this includes slave-server-replies.
- *NIS clients*. NIS clients, like most Windows NT workstations, authenticate against the NIS server (or the Windows NT domain controller in the Windows NT workstations case) to log on.

25.4.4. Using NIS/YP

This section will deal with setting up a sample NIS environment.

Notatka: This section assumes that you are running FreeBSD 3.3 or later. The instructions given here will *probably* work for any version of FreeBSD greater than 3.0, but there are no guarantees that this is true.

25.4.4.1. Planning

Let us assume that you are the administrator of a small university lab. This lab, which consists of 15 FreeBSD machines, currently has no centralized point of administration; each machine has its own `/etc/passwd` and `/etc/master.passwd`. These files are kept in sync with each other only through manual intervention; currently, when you add a user to the lab, you must run `adduser` on all 15 machines. Clearly, this has to change, so you have decided to convert the lab to use NIS, using two of the machines as servers.

Therefore, the configuration of the lab now looks something like:

Machine name	IP address	Machine role
ellington	10.0.0.2	NIS master
coltrane	10.0.0.3	NIS slave
basie	10.0.0.4	Faculty workstation
bird	10.0.0.5	Client machine
cli[1-11]	10.0.0.[6-17]	Other client machines

If you are setting up a NIS scheme for the first time, it is a good idea to think through how you want to go about it. No matter what the size of your network, there are a few decisions that need to be made.

25.4.4.1.1. Choosing a NIS Domain Name

This might not be the „domainname” that you are used to. It is more accurately called the „NIS domainname”. When a client broadcasts its requests for info, it includes the name of the NIS domain that it is part of. This is how multiple servers on one network can tell which server should answer which request. Think of the NIS domainname as the name for a group of hosts that are related in some way.

Some organizations choose to use their Internet domainname for their NIS domainname. This is not recommended as it can cause confusion when trying to debug network problems. The NIS domainname should be unique within your network and it is helpful if it describes the group of machines it represents. For example, the Art department at Acme Inc. might be in the „acme-art” NIS domain. For this example, assume you have chosen the name `test-domain`.

However, some operating systems (notably SunOS) use their NIS domain name as their Internet domain name. If one or more machines on your network have this restriction, you *must* use the Internet domain name as your NIS domain name.

25.4.4.1.2. Physical Server Requirements

There are several things to keep in mind when choosing a machine to use as a NIS server. One of the unfortunate things about NIS is the level of dependency the clients have on the server. If a client cannot contact the server for its NIS domain, very often the machine becomes unusable. The lack of user and group information causes most systems to temporarily freeze up. With this in mind you should make sure to choose a machine that will not be prone to being rebooted regularly, or one that might be used for development. The NIS server should ideally be a stand alone machine whose sole purpose in life is to be an NIS server. If you have a network that is not very heavily used, it is acceptable to put the NIS server on a machine running other services, just keep in mind that if the NIS server becomes unavailable, it will affect *all* of your NIS clients adversely.

25.4.4.2. NIS Servers

The canonical copies of all NIS information are stored on a single machine called the NIS master server. The databases used to store the information are called NIS maps. In FreeBSD, these maps are stored in `/var/yp/[domainname]` where `[domainname]` is the name of the NIS domain being served. A single NIS server can support several domains at once, therefore it is possible to have several such directories, one for each supported domain. Each domain will have its own independent set of maps.

NIS master and slave servers handle all NIS requests with the `ypserv` daemon. `ypserv` is responsible for receiving incoming requests from NIS clients, translating the requested domain and map name to a path to the corresponding database file and transmitting data from the database back to the client.

25.4.4.2.1. Setting Up a NIS Master Server

Setting up a master NIS server can be relatively straight forward, depending on your needs. FreeBSD comes with support for NIS out-of-the-box. All you need is to add the following lines to `/etc/rc.conf`, and FreeBSD will do the rest for you.

1.

```
nisdomainname="test-domain"
```

This line will set the NIS domainname to `test-domain` upon network setup (e.g. after reboot).

2.

```
nis_server_enable="YES"
```

This will tell FreeBSD to start up the NIS server processes when the networking is next brought up.

3.

```
nis_yppasswdd_enable="YES"
```

This will enable the `rpc.yppasswdd` daemon which, as mentioned above, will allow users to change their NIS password from a client machine.

Notatka: Depending on your NIS setup, you may need to add further entries. See the section about NIS servers that are also NIS clients, below, for details.

Now, all you have to do is to run the command `/etc/netstart` as superuser. It will set up everything for you, using the values you defined in `/etc/rc.conf`.

25.4.4.2.2. Initializing the NIS Maps

The *NIS maps* are database files, that are kept in the `/var/yp` directory. They are generated from configuration files in the `/etc` directory of the NIS master, with one exception: the `/etc/master.passwd` file. This is for a good reason, you do not want to propagate passwords to your `root` and other administrative accounts to all the servers in the NIS domain. Therefore, before we initialize the NIS maps, you should:

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
# vi master.passwd
```

You should remove all entries regarding system accounts (`bin`, `tty`, `kmem`, `games`, etc), as well as any accounts that you do not want to be propagated to the NIS clients (for example `root` and any other UID 0 (superuser) accounts).

Notatka: Make sure the `/var/yp/master.passwd` is neither group nor world readable (mode 600)! Use the `chmod` command, if appropriate.

When you have finished, it is time to initialize the NIS maps! FreeBSD includes a script named `ypinit` to do this for you (see its manual page for more information). Note that this script is available on most UNIX Operating Systems, but not on all. On Digital UNIX/Compaq Tru64 UNIX it is called `ypsetup`. Because we are generating maps for an NIS master, we are going to pass the `-m` option to `ypinit`. To generate the NIS maps, assuming you already performed the steps above, run:

```
ellington# ypinit -m test-domain
Server Type: MASTER Domain: test-domain
Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
At this point, we have to construct a list of this domains YP servers.
rod.darktech.org is already known as master server.
```

Please continue to add any slave servers, one per line. When you are done with the list, type a <control D>.

```
master server   : ellington
```

```
next host to add: coltrane
```

```
next host to add: ^D
```

The current list of NIS servers looks like this:

```
ellington
```

```
coltrane
```

```
Is this correct? [y/n: y] y
```

```
[..output from map generation..]
```

NIS Map update completed.

ellington has been setup as an YP master server without any errors.

ypinit should have created `/var/yp/Makefile` from `/var/yp/Makefile.dist`. When created, this file assumes that you are operating in a single server NIS environment with only FreeBSD machines. Since `test-domain` has a slave server as well, you must edit `/var/yp/Makefile`:

```
ellington# vi /var/yp/Makefile
```

You should comment out the line that says

```
NOPUSH = "True"
```

(if it is not commented out already).

25.4.4.2.3. Setting up a NIS Slave Server

Setting up an NIS slave server is even more simple than setting up the master. Log on to the slave server and edit the file `/etc/rc.conf` as you did before. The only difference is that we now must use the `-s` option when running `ypinit`. The `-s` option requires the name of the NIS master be passed to it as well, so our command line looks like:

```
coltrane# ypinit -s ellington test-domain
```

```
Server Type: SLAVE Domain: test-domain Master: ellington
```

Creating an YP server will require that you answer a few questions. Questions will all be asked at the beginning of the procedure.

```
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
```

Ok, please remember to go back and redo manually whatever fails.

If you don't, something might not work.

There will be no further questions. The remainder of the procedure should take a few minutes, to copy the databases from ellington.

```
Transferring netgroup...
```

```
ypxfr: Exiting: Map successfully transferred
```

```
Transferring netgroup.byuser...
```

```
ypxfr: Exiting: Map successfully transferred
```

```
Transferring netgroup.byhost...
```

```
ypxfr: Exiting: Map successfully transferred
```

```

Transferring master.passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring group.bygid...
ypxfr: Exiting: Map successfully transferred
Transferring group.byname...
ypxfr: Exiting: Map successfully transferred
Transferring services.byname...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.byname...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.byname...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring netid.byname...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring ypservers...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byname...
ypxfr: Exiting: Map successfully transferred

```

coltrane has been setup as an YP slave server without any errors.
Don't forget to update map ypservers on ellington.

You should now have a directory called `/var/yp/test-domain`. Copies of the NIS master server's maps should be in this directory. You will need to make sure that these stay updated. The following `/etc/crontab` entries on your slave servers should do the job:

```

20      *      *      *      *      root    /usr/libexec/ypxfr passwd.byname
21      *      *      *      *      root    /usr/libexec/ypxfr passwd.byuid

```

These two lines force the slave to sync its maps with the maps on the master server. Although these entries are not mandatory, since the master server attempts to ensure any changes to its NIS maps are communicated to its slaves and because password information is vital to systems depending on the server, it is a good idea to force the updates. This is more important on busy networks where map updates might not always complete.

Now, run the command `/etc/netstart` on the slave server as well, which again starts the NIS server.

25.4.4.3. NIS Clients

An NIS client establishes what is called a binding to a particular NIS server using the `ypbind` daemon. `ypbind` checks the system's default domain (as set by the `domainname` command), and begins broadcasting RPC requests on the local network. These requests specify the name of the domain for which `ypbind` is attempting to establish a binding. If a server that has been configured to serve the requested domain receives one of the broadcasts, it will respond to `ypbind`, which will record the server's address. If there are several servers available (a master and several slaves, for example), `ypbind` will use the address of the first one to respond. From that point on, the client system will direct all of its NIS requests to that server. `ypbind` will occasionally „ping” the server to make sure it is still up and running. If it fails to receive a reply to one of its pings within a reasonable amount of time, `ypbind` will mark the domain as unbound and begin broadcasting again in the hopes of locating another server.

25.4.4.3.1. Setting Up a NIS Client

Setting up a FreeBSD machine to be a NIS client is fairly straightforward.

1. Edit the file `/etc/rc.conf` and add the following lines in order to set the NIS domainname and start `ypbind` upon network startup:

```
nisdomainname="test-domain"
nis_client_enable="YES"
```

2. To import all possible password entries from the NIS server, remove all user accounts from your `/etc/master.passwd` file and use `vipw` to add the following line to the end of the file:

```
+:::~:::
```

Notatka: This line will afford anyone with a valid account in the NIS server's password maps an account. There are many ways to configure your NIS client by changing this line. See the `netgroups` section below for more information. For more detailed reading see O'Reilly's book on `Managing NFS and NIS`.

Notatka: You should keep at least one local account (i.e. not imported via NIS) in your `/etc/master.passwd` and this account should also be a member of the group `wheel`. If there is something wrong with NIS, this account can be used to log in remotely, become `root`, and fix things.

3. To import all possible group entries from the NIS server, add this line to your `/etc/group` file:

```
+:::~:::
```

After completing these steps, you should be able to run `ypcat passwd` and see the NIS server's `passwd` map.

25.4.5. NIS Security

In general, any remote user can issue an RPC to `ypserv(8)` and retrieve the contents of your NIS maps, provided the remote user knows your domainname. To prevent such unauthorized transactions, `ypserv(8)` supports a feature called „`securenets`” which can be used to restrict access to a given set of hosts. At startup, `ypserv(8)` will attempt to load the `securenets` information from a file called `/var/yp/securenets`.

Notatka: This path varies depending on the path specified with the `-p` option. This file contains entries that consist of a network specification and a network mask separated by white space. Lines starting with „#” are considered to be comments. A sample `securenets` file might look like this:

```
# allow connections from local host -- mandatory
127.0.0.1      255.255.255.255
# allow connections from any host
# on the 192.168.128.0 network
192.168.128.0 255.255.255.0
# allow connections from any host
# between 10.0.0.0 to 10.0.15.255
# this includes the machines in the testlab
10.0.0.0       255.255.240.0
```

If `ypserv(8)` receives a request from an address that matches one of these rules, it will process the request normally. If the address fails to match a rule, the request will be ignored and a warning message will be logged. If the `/var/yp/securenets` file does not exist, `ypserv` will allow connections from any host.

The `ypserv` program also has support for Wietse Venema’s **TCP Wrapper** package. This allows the administrator to use the **TCP Wrapper** configuration files for access control instead of `/var/yp/securenets`.

Notatka: While both of these access control mechanisms provide some security, they, like the privileged port test, are vulnerable to „IP spoofing” attacks. All NIS-related traffic should be blocked at your firewall.

Servers using `/var/yp/securenets` may fail to serve legitimate NIS clients with archaic TCP/IP implementations. Some of these implementations set all host bits to zero when doing broadcasts and/or fail to observe the subnet mask when calculating the broadcast address. While some of these problems can be fixed by changing the client configuration, other problems may force the retirement of the client systems in question or the abandonment of `/var/yp/securenets`.

Using `/var/yp/securenets` on a server with such an archaic implementation of TCP/IP is a really bad idea and will lead to loss of NIS functionality for large parts of your network.

The use of the **TCP Wrapper** package increases the latency of your NIS server. The additional delay may be long enough to cause timeouts in client programs, especially in busy networks or with slow NIS servers. If one or more of your client systems suffers from these symptoms, you should convert the client systems in question into NIS slave servers and force them to bind to themselves.

25.4.6. Barring Some Users from Logging On

In our lab, there is a machine `basie` that is supposed to be a faculty only workstation. We do not want to take this machine out of the NIS domain, yet the `passwd` file on the master NIS server contains accounts for both faculty and students. What can we do?

There is a way to bar specific users from logging on to a machine, even if they are present in the NIS database. To do this, all you must do is add `-username` to the end of the `/etc/master.passwd` file on the client machine, where `username` is the username of the user you wish to bar from logging in. This should preferably be done using `vipw`, since `vipw` will sanity check your changes to `/etc/master.passwd`, as well as automatically rebuild the password database when you finish editing. For example, if we wanted to bar user `bill` from logging on to `basie` we would:

```

basie# vipw
[add -bill to the end, exit]
vipw: rebuilding the database...
vipw: done

basie# cat /etc/master.passwd

root:[password]:0:0::0:0:The super-user:/root:/bin/csh
toor:[password]:0:0::0:0:The other super-user:/root:/bin/sh
daemon:*:1:1::0:0:Owner of many system processes:/root:/sbin/nologin
operator:*:2:5::0:0:System &:/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source,,:/sbin/nologin
tty:*:4:65533::0:0:Tty Sandbox:/sbin/nologin
kmem:*:5:65533::0:0:KMem Sandbox:/sbin/nologin
games:*:7:13::0:0:Games pseudo-user:/usr/games:/sbin/nologin
news:*:8:8::0:0:News Subsystem:/sbin/nologin
man:*:9:9::0:0:Mister Man Pages:/usr/share/man:/sbin/nologin
bind:*:53:53::0:0:Bind Sandbox:/sbin/nologin
uucp:*:66:66::0:0:UUCP pseudo-user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
xten:*:67:67::0:0:X-10 daemon:/usr/local/xten:/sbin/nologin
pop:*:68:6::0:0:Post Office Owner:/nonexistent:/sbin/nologin
nobody:*:65534:65534::0:0:Unprivileged user:/nonexistent:/sbin/nologin
+:::
-bill

basie#

```

25.4.7. Using Netgroups

Contributed by Udo Erdelhoff.

The method shown in the previous section works reasonably well if you need special rules for a very small number of users and/or machines. On larger networks, you *will* forget to bar some users from logging onto sensitive machines, or you may even have to modify each machine separately, thus losing the main benefit of NIS: *centralized* administration.

The NIS developers' solution for this problem is called *netgroups*. Their purpose and semantics can be compared to the normal groups used by UNIX file systems. The main differences are the lack of a numeric ID and the ability to define a netgroup by including both user accounts and other netgroups.

Netgroups were developed to handle large, complex networks with hundreds of users and machines. On one hand, this is a Good Thing if you are forced to deal with such a situation. On the other hand, this complexity makes it almost impossible to explain netgroups with really simple examples. The example used in the remainder of this section demonstrates this problem.

Let us assume that your successful introduction of NIS in your laboratory caught your superiors' interest. Your next job is to extend your NIS domain to cover some of the other machines on campus. The two tables contain the names of the new users and new machines as well as brief descriptions of them.

User Name(s)	Description
alpha,beta	Normal employees of the IT department

User Name(s)

charlie, delta
 echo, foxtrott, golf, ...
 able, baker, ...

Description

The new apprentices of the IT department
 Ordinary employees
 The current interns

Machine Name(s)

war, death, famine, pollution
 pride, greed, envy, wrath, lust, sloth
 one, two, three, four, ...
 trashcan

Description

Your most important servers. Only the IT employees are allowed to log onto these machines.
 Less important servers. All members of the IT department are allowed to login onto these machines.
 Ordinary workstations. Only the *real* employees are allowed to use these machines.
 A very old machine without any critical data. Even the intern is allowed to use this box.

If you tried to implement these restrictions by separately blocking each user, you would have to add one `-user` line to each system's `passwd` for each user who is not allowed to login onto that system. If you forget just one entry, you could be in trouble. It may be feasible to do this correctly during the initial setup, however you *will* eventually forget to add the lines for new users during day-to-day operations. After all, Murphy was an optimist.

Handling this situation with netgroups offers several advantages. Each user need not be handled separately; you assign a user to one or more netgroups and allow or forbid logins for all members of the netgroup. If you add a new machine, you will only have to define login restrictions for netgroups. If a new user is added, you will only have to add the user to one or more netgroups. Those changes are independent of each other: no more „for each combination of user and machine do...” If your NIS setup is planned carefully, you will only have to modify exactly one central configuration file to grant or deny access to machines.

The first step is the initialization of the NIS map netgroup. FreeBSD's `ypinit(8)` does not create this map by default, but its NIS implementation will support it once it has been created. To create an empty map, simply type

```
ellington# vi /var/yp/netgroup
```

and start adding content. For our example, we need at least four netgroups: IT employees, IT apprentices, normal employees and interns.

```
IT_EMP  ( ,alpha,test-domain)    ( ,beta,test-domain)
IT_APP  ( ,charlie,test-domain)  ( ,delta,test-domain)
USERS   ( ,echo,test-domain)     ( ,foxtrott,test-domain) \
        ( ,golf,test-domain)
INTERNS ( ,able,test-domain)     ( ,baker,test-domain)
```

`IT_EMP`, `IT_APP` etc. are the names of the netgroups. Each bracketed group adds one or more user accounts to it. The three fields inside a group are:

1. The name of the host(s) where the following items are valid. If you do not specify a hostname, the entry is valid on all hosts. If you do specify a hostname, you will enter a realm of darkness, horror and utter confusion.
2. The name of the account that belongs to this netgroup.

3. The NIS domain for the account. You can import accounts from other NIS domains into your netgroup if you are one of the unlucky fellows with more than one NIS domain.

Each of these fields can contain wildcards. See `netgroup(5)` for details.

Notatka: Netgroup names longer than 8 characters should not be used, especially if you have machines running other operating systems within your NIS domain. The names are case sensitive; using capital letters for your netgroup names is an easy way to distinguish between user, machine and netgroup names.

Some NIS clients (other than FreeBSD) cannot handle netgroups with a large number of entries. For example, some older versions of SunOS start to cause trouble if a netgroup contains more than 15 *entries*. You can circumvent this limit by creating several sub-netgroups with 15 users or less and a real netgroup that consists of the sub-netgroups:

```
BIGGRP1  (,joe1,domain) (,joe2,domain) (,joe3,domain) [...]
BIGGRP2  (,joe16,domain) (,joe17,domain) [...]
BIGGRP3  (,joe31,domain) (,joe32,domain)
BIGGROUP  BIGGRP1 BIGGRP2 BIGGRP3
```

You can repeat this process if you need more than 225 users within a single netgroup.

Activating and distributing your new NIS map is easy:

```
ellington# cd /var/yp
ellington# make
```

This will generate the three NIS maps `netgroup`, `netgroup.byhost` and `netgroup.byuser`. Use `ypcat(1)` to check if your new NIS maps are available:

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

The output of the first command should resemble the contents of `/var/yp/netgroup`. The second command will not produce output if you have not specified host-specific netgroups. The third command can be used to get the list of netgroups for a user.

The client setup is quite simple. To configure the server `war`, you only have to start `vipw(8)` and replace the line

```
+:::~:::
```

with

```
+@IT_EMP:::~:::
```

Now, only the data for the users defined in the netgroup `IT_EMP` is imported into `war`'s password database and only these users are allowed to login.

Unfortunately, this limitation also applies to the `~` function of the shell and all routines converting between user names and numerical user IDs. In other words, `cd ~user` will not work, `ls -l` will show the numerical ID instead of the username and `find . -user joe -print` will fail with `No such user`. To fix this, you will have to import all user entries *without allowing them to login onto your servers*.

This can be achieved by adding another line to `/etc/master.passwd`. This line should contain:

+:::/:sbin/nologin, meaning „Import all entries but replace the shell with /sbin/nologin in the imported entries”. You can replace any field in the passwd entry by placing a default value in your /etc/master.passwd.

Ostrzeżenie Make sure that the line +:::/:sbin/nologin is placed after +@IT_EMP:::~. Otherwise, all user accounts imported from NIS will have /sbin/nologin as their login shell.

After this change, you will only have to change one NIS map if a new employee joins the IT department. You could use a similar approach for the less important servers by replacing the old +:::~ in their local version of /etc/master.passwd with something like this:

```
+@IT_EMP:::~
+@IT_APP:::~
+:::/:sbin/nologin
```

The corresponding lines for the normal workstations could be:

```
+@IT_EMP:::~
+@USERS:::~
+:::/:sbin/nologin
```

And everything would be fine until there is a policy change a few weeks later: The IT department starts hiring interns. The IT interns are allowed to use the normal workstations and the less important servers; and the IT apprentices are allowed to login onto the main servers. You add a new netgroup IT_INTERN, add the new IT interns to this netgroup and start to change the configuration on each and every machine... As the old saying goes: „Errors in centralized planning lead to global mess”.

NIS’ ability to create netgroups from other netgroups can be used to prevent situations like these. One possibility is the creation of role-based netgroups. For example, you could create a netgroup called BIGSRV to define the login restrictions for the important servers, another netgroup called SMALLSRV for the less important servers and a third netgroup called USERBOX for the normal workstations. Each of these netgroups contains the netgroups that are allowed to login onto these machines. The new entries for your NIS map netgroup should look like this:

```
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP  ITINTERN
USERBOX   IT_EMP  ITINTERN  USERS
```

This method of defining login restrictions works reasonably well if you can define groups of machines with identical restrictions. Unfortunately, this is the exception and not the rule. Most of the time, you will need the ability to define login restrictions on a per-machine basis.

Machine-specific netgroup definitions are the other possibility to deal with the policy change outlined above. In this scenario, the /etc/master.passwd of each box contains two lines starting with „+”. The first of them adds a netgroup with the accounts allowed to login onto this machine, the second one adds all other accounts with /sbin/nologin as shell. It is a good idea to use the „ALL-CAPS” version of the machine name as the name of the netgroup. In other words, the lines should look like this:

```
+@BOXNAME:::~
+:::/:sbin/nologin
```

Once you have completed this task for all your machines, you will not have to modify the local versions of `/etc/master.passwd` ever again. All further changes can be handled by modifying the NIS map. Here is an example of a possible netgroup map for this scenario with some additional goodies:

```
# Define groups of users first
IT_EMP      (,alpha,test-domain)      (,beta,test-domain)
IT_APP      (,charlie,test-domain)    (,delta,test-domain)
DEPT1       (,echo,test-domain)       (,foxtrott,test-domain)
DEPT2       (,golf,test-domain)       (,hotel,test-domain)
DEPT3       (,india,test-domain)      (,juliet,test-domain)
ITINTERN    (,kilo,test-domain)      (,lima,test-domain)
D_INTERNS   (,able,test-domain)      (,baker,test-domain)
#
# Now, define some groups based on roles
USERS       DEPT1    DEPT2    DEPT3
BIGSRV      IT_EMP   IT_APP
SMALLSRV     IT_EMP   IT_APP   ITINTERN
USERBOX     IT_EMP   ITINTERN  USERS
#
# And a groups for a special tasks
# Allow echo and golf to access our anti-virus-machine
SECURITY    IT_EMP   (,echo,test-domain) (,golf,test-domain)
#
# machine-based netgroups
# Our main servers
WAR          BIGSRV
FAMINE       BIGSRV
# User india needs access to this server
POLLUTION   BIGSRV   (,india,test-domain)
#
# This one is really important and needs more access restrictions
DEATH       IT_EMP
#
# The anti-virus-machine mentioned above
ONE         SECURITY
#
# Restrict a machine to a single user
TWO         (,hotel,test-domain)
# [...more groups to follow]
```

If you are using some kind of database to manage your user accounts, you should be able to create the first part of the map with your database's report tools. This way, new users will automatically have access to the boxes.

One last word of caution: It may not always be advisable to use machine-based netgroups. If you are deploying a couple of dozen or even hundreds of identical machines for student labs, you should use role-based netgroups instead of machine-based netgroups to keep the size of the NIS map within reasonable limits.

25.4.8. Important Things to Remember

There are still a couple of things that you will need to do differently now that you are in an NIS environment.

- Every time you wish to add a user to the lab, you must add it to the master NIS server *only*, and *you must remember to rebuild the NIS maps*. If you forget to do this, the new user will not be able to login anywhere except on the NIS master. For example, if we needed to add a new user `jsmith` to the lab, we would:

```
# pw useradd jsmith
# cd /var/yp
# make test-domain
```

You could also run `adduser jsmith` instead of `pw useradd jsmith`.

- *Keep the administration accounts out of the NIS maps.* You do not want to be propagating administrative accounts and passwords to machines that will have users that should not have access to those accounts.
- *Keep the NIS master and slave secure, and minimize their downtime.* If somebody either hacks or simply turns off these machines, they have effectively rendered many people without the ability to login to the lab.

This is the chief weakness of any centralized administration system. If you do not protect your NIS servers, you will have a lot of angry users!

25.4.9. NIS v1 Compatibility

FreeBSD's **ypserv** has some support for serving NIS v1 clients. FreeBSD's NIS implementation only uses the NIS v2 protocol, however other implementations include support for the v1 protocol for backwards compatibility with older systems. The **ypbind** daemons supplied with these systems will try to establish a binding to an NIS v1 server even though they may never actually need it (and they may persist in broadcasting in search of one even after they receive a response from a v2 server). Note that while support for normal client calls is provided, this version of **ypserv** does not handle v1 map transfer requests; consequently, it cannot be used as a master or slave in conjunction with older NIS servers that only support the v1 protocol. Fortunately, there probably are not any such servers still in use today.

25.4.10. NIS Servers That Are Also NIS Clients

Care must be taken when running **ypserv** in a multi-server domain where the server machines are also NIS clients. It is generally a good idea to force the servers to bind to themselves rather than allowing them to broadcast bind requests and possibly become bound to each other. Strange failure modes can result if one server goes down and others are dependent upon it. Eventually all the clients will time out and attempt to bind to other servers, but the delay involved can be considerable and the failure mode is still present since the servers might bind to each other all over again.

You can force a host to bind to a particular server by running **ypbind** with the `-S` flag. If you do not want to do this manually each time you reboot your NIS server, you can add the following lines to your `/etc/rc.conf`:

```
nis_client_enable="YES" # run client stuff as well
nis_client_flags="-S NIS domain,server"
```

See **ypbind(8)** for further information.

25.4.11. Password Formats

One of the most common issues that people run into when trying to implement NIS is password format compatibility. If your NIS server is using DES encrypted passwords, it will only support clients that are also using DES. For

example, if you have Solaris NIS clients in your network, then you will almost certainly need to use DES encrypted passwords.

To check which format your servers and clients are using, look at `/etc/login.conf`. If the host is configured to use DES encrypted passwords, then the `default` class will contain an entry like this:

```
default:\
    :passwd_format=des:\
    :copyright=/etc/COPYRIGHT:\
    [Further entries elided]
```

Other possible values for the `passwd_format` capability include `blf` and `md5` (for Blowfish and MD5 encrypted passwords, respectively).

If you have made changes to `/etc/login.conf`, you will also need to rebuild the login capability database, which is achieved by running the following command as `root`:

```
# cap_mkdb /etc/login.conf
```

Notatka: The format of passwords already in `/etc/master.passwd` will not be updated until a user changes his password for the first time *after* the login capability database is rebuilt.

Next, in order to ensure that passwords are encrypted with the format that you have chosen, you should also check that the `crypt_default` in `/etc/auth.conf` gives precedence to your chosen password format. To do this, place the format that you have chosen first in the list. For example, when using DES encrypted passwords, the entry would be:

```
crypt_default    =    des blf md5
```

Having followed the above steps on each of the FreeBSD based NIS servers and clients, you can be sure that they all agree on which password format is used within your network. If you have trouble authenticating on an NIS client, this is a pretty good place to start looking for possible problems. Remember: if you want to deploy an NIS server for a heterogenous network, you will probably have to use DES on all systems because it is the lowest common standard.

25.5. Automatic Network Configuration (DHCP)

Written by Greg Sutter.

25.5.1. What Is DHCP?

DHCP, the Dynamic Host Configuration Protocol, describes the means by which a system can connect to a network and obtain the necessary information for communication upon that network. FreeBSD versions prior to 6.0 use the ISC (Internet Software Consortium) DHCP client (`dhclient(8)`) implementation. Later versions use the OpenBSD `dhclient` taken from OpenBSD 3.7. All information here regarding `dhclient` is for use with either of the ISC or OpenBSD DHCP clients. The DHCP server is the one included in the ISC distribution.

25.5.2. What This Section Covers

This section describes both the client-side components of the ISC and OpenBSD DHCP client and server-side components of the ISC DHCP system. The client-side program, `dhclient`, comes integrated within FreeBSD, and the server-side portion is available from the `net/isc-dhcp3-server` port. The `dhclient(8)`, `dhcp-options(5)`, and `dhclient.conf(5)` manual pages, in addition to the references below, are useful resources.

25.5.3. How It Works

When `dhclient`, the DHCP client, is executed on the client machine, it begins broadcasting requests for configuration information. By default, these requests are on UDP port 68. The server replies on UDP 67, giving the client an IP address and other relevant network information such as netmask, router, and DNS servers. All of this information comes in the form of a DHCP „lease” and is only valid for a certain time (configured by the DHCP server maintainer). In this manner, stale IP addresses for clients no longer connected to the network can be automatically reclaimed.

DHCP clients can obtain a great deal of information from the server. An exhaustive list may be found in `dhcp-options(5)`.

25.5.4. FreeBSD Integration

FreeBSD fully integrates the ISC or OpenBSD DHCP client, `dhclient` (according to the FreeBSD version you run). DHCP client support is provided within both the installer and the base system, obviating the need for detailed knowledge of network configurations on any network that runs a DHCP server. `dhclient` has been included in all FreeBSD distributions since 3.2.

DHCP is supported by `sysinstall`. When configuring a network interface within `sysinstall`, the second question asked is: „Do you want to try DHCP configuration of the interface?”. Answering affirmatively will execute `dhclient`, and if successful, will fill in the network configuration information automatically.

There are two things you must do to have your system use DHCP upon startup:

- Make sure that the `bpf` device is compiled into your kernel. To do this, add `device bpf` to your kernel configuration file, and rebuild the kernel. For more information about building kernels, see Rozdział 8.

The `bpf` device is already part of the `GENERIC` kernel that is supplied with FreeBSD, so if you do not have a custom kernel, you should not need to create one in order to get DHCP working.

Notatka: For those who are particularly security conscious, you should be warned that `bpf` is also the device that allows packet sniffers to work correctly (although they still have to be run as `root`). `bpf` is required to use DHCP, but if you are very sensitive about security, you probably should not add `bpf` to your kernel in the expectation that at some point in the future you will be using DHCP.

- Edit your `/etc/rc.conf` to include the following:

```
ifconfig_fxp0="DHCP"
```

Notatka: Be sure to replace `fxp0` with the designation for the interface that you wish to dynamically configure, as described in Sekcja 11.8.

If you are using a different location for `dhclient`, or if you wish to pass additional flags to `dhclient`, also include the following (editing as necessary):

```
dhcp_program="/sbin/dhclient"
dhcp_flags=""
```

The DHCP server, **dhcpcd**, is included as part of the `net/isc-dhcp3-server` port in the ports collection. This port contains the ISC DHCP server and documentation.

25.5.5. Files

- `/etc/dhclient.conf`

`dhclient` requires a configuration file, `/etc/dhclient.conf`. Typically the file contains only comments, the defaults being reasonably sane. This configuration file is described by the `dhclient.conf(5)` manual page.

- `/sbin/dhclient`

`dhclient` is statically linked and resides in `/sbin`. The `dhclient(8)` manual page gives more information about `dhclient`.

- `/sbin/dhclient-script`

`dhclient-script` is the FreeBSD-specific DHCP client configuration script. It is described in `dhclient-script(8)`, but should not need any user modification to function properly.

- `/var/db/dhclient.leases`

The DHCP client keeps a database of valid leases in this file, which is written as a log. `dhclient.leases(5)` gives a slightly longer description.

25.5.6. Further Reading

The DHCP protocol is fully described in RFC 2131 (<http://www.freessoft.org/CIE/RFC/2131/>). An informational resource has also been set up at <http://www.dhcp.org/>.

25.5.7. Installing and Configuring a DHCP Server

25.5.7.1. What This Section Covers

This section provides information on how to configure a FreeBSD system to act as a DHCP server using the ISC (Internet Software Consortium) implementation of the DHCP server.

The server is not provided as part of FreeBSD, and so you will need to install the `net/isc-dhcp3-server` port to provide this service. See Rozdział 4 for more information on using the Ports Collection.

25.5.7.2. DHCP Server Installation

In order to configure your FreeBSD system as a DHCP server, you will need to ensure that the `bpf(4)` device is compiled into your kernel. To do this, add `device bpf` to your kernel configuration file, and rebuild the kernel. For more information about building kernels, see [Rozdział 8](#).

The `bpf` device is already part of the `GENERIC` kernel that is supplied with FreeBSD, so you do not need to create a custom kernel in order to get DHCP working.

Notatka: Those who are particularly security conscious should note that `bpf` is also the device that allows packet sniffers to work correctly (although such programs still need privileged access). `bpf` is required to use DHCP, but if you are very sensitive about security, you probably should not include `bpf` in your kernel purely because you expect to use DHCP at some point in the future.

The next thing that you will need to do is edit the sample `dhcpd.conf` which was installed by the `net/isc-dhcp3-server` port. By default, this will be `/usr/local/etc/dhcpd.conf.sample`, and you should copy this to `/usr/local/etc/dhcpd.conf` before proceeding to make changes.

25.5.7.3. Configuring the DHCP Server

`dhcpd.conf` is comprised of declarations regarding subnets and hosts, and is perhaps most easily explained using an example :

```
option domain-name "example.com";❶
option domain-name-servers 192.168.4.100;❷
option subnet-mask 255.255.255.0;❸

default-lease-time 3600;❹
max-lease-time 86400;❺
ddns-update-style none;❻

subnet 192.168.4.0 netmask 255.255.255.0 {
    range 192.168.4.129 192.168.4.254;❼
    option routers 192.168.4.1;❽
}

host mailhost {
    hardware ethernet 02:03:04:05:06:07;❾
    fixed-address mailhost.example.com;(10)
}
```

- ❶ This option specifies the domain that will be provided to clients as the default search domain. See `resolv.conf(5)` for more information on what this means.
- ❷ This option specifies a comma separated list of DNS servers that the client should use.
- ❸ The netmask that will be provided to clients.
- ❹ A client may request a specific length of time that a lease will be valid. Otherwise the server will assign a lease with this expiry value (in seconds).

- ⑤ This is the maximum length of time that the server will lease for. Should a client request a longer lease, a lease will be issued, although it will only be valid for `max-lease-time` seconds.
- ⑥ This option specifies whether the DHCP server should attempt to update DNS when a lease is accepted or released. In the ISC implementation, this option is *required*.
- ⑦ This denotes which IP addresses should be used in the pool reserved for allocating to clients. IP addresses between, and including, the ones stated are handed out to clients.
- ⑧ Declares the default gateway that will be provided to clients.
- ⑨ The hardware MAC address of a host (so that the DHCP server can recognize a host when it makes a request).
- (10) Specifies that the host should always be given the same IP address. Note that using a hostname is correct here, since the DHCP server will resolve the hostname itself before returning the lease information.

Once you have finished writing your `dhcpd.conf`, you should enable the DHCP server in `/etc/rc.conf`, i.e. by adding:

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

Replace the `dc0` interface name with the interface (or interfaces, separated by whitespace) that your DHCP server should listen on for DHCP client requests.

Then, you can proceed to start the server by issuing the following command:

```
# /usr/local/etc/rc.d/isc-dhcpd.sh start
```

Should you need to make changes to the configuration of your server in the future, it is important to note that sending a `SIGHUP` signal to **dhcpd** does *not* result in the configuration being reloaded, as it does with most daemons. You will need to send a `SIGTERM` signal to stop the process, and then restart it using the command above.

25.5.7.4. Files

- `/usr/local/sbin/dhcpd`

dhcpd is statically linked and resides in `/usr/local/sbin`. The `dhcpd(8)` manual page installed with the port gives more information about **dhcpd**.

- `/usr/local/etc/dhcpd.conf`

dhcpd requires a configuration file, `/usr/local/etc/dhcpd.conf` before it will start providing service to clients. This file needs to contain all the information that should be provided to clients that are being serviced, along with information regarding the operation of the server. This configuration file is described by the `dhcpd.conf(5)` manual page installed by the port.

- `/var/db/dhcpd.leases`

The DHCP server keeps a database of leases it has issued in this file, which is written as a log. The manual page `dhcpd.leases(5)`, installed by the port gives a slightly longer description.

- `/usr/local/sbin/dhcrelay`

dhcrelay is used in advanced environments where one DHCP server forwards a request from a client to another DHCP server on a separate network. If you require this functionality, then install the `net/isc-dhcp3-relay` port. The `dhcrelay(8)` manual page provided with the port contains more detail.

25.6. Domain Name System (DNS)

Contributed by Chern Lee, Tom Rhodes, i Daniel Gerzo.

25.6.1. Overview

FreeBSD utilizes, by default, a version of BIND (Berkeley Internet Name Domain), which is the most common implementation of the DNS protocol. DNS is the protocol through which names are mapped to IP addresses, and vice versa. For example, a query for `www.FreeBSD.org` will receive a reply with the IP address of The FreeBSD Project's web server, whereas, a query for `ftp.FreeBSD.org` will return the IP address of the corresponding FTP machine. Likewise, the opposite can happen. A query for an IP address can resolve its hostname. It is not necessary to run a name server to perform DNS lookups on a system.

FreeBSD currently comes with BIND9 DNS server software by default. Our installation provides enhanced security features, a new file system layout and automated `chroot(8)` configuration.

DNS is coordinated across the Internet through a somewhat complex system of authoritative root, Top Level Domain (TLD), and other smaller-scale name servers which host and cache individual domain information.

Currently, BIND is maintained by the Internet Software Consortium <http://www.isc.org/>.

25.6.2. Terminology

To understand this document, some terms related to DNS must be understood.

Term	Definition
Forward DNS	Mapping of hostnames to IP addresses.
Origin	Refers to the domain covered in a particular zone file.
named , BIND, name server	Common names for the BIND name server package within FreeBSD.
Resolver	A system process through which a machine queries a name server for zone information.
Reverse DNS	The opposite of forward DNS; mapping of IP addresses to hostnames.
Root zone	The beginning of the Internet zone hierarchy. All zones fall under the root zone, similar to how all files in a file system fall under the root directory.
Zone	An individual domain, subdomain, or portion of the DNS administered by the same authority.

Examples of zones:

- `.` is the root zone.
- `org.` is a Top Level Domain (TLD) under the root zone.

- `example.org.` is a zone under the `org.` TLD.
- `1.168.192.in-addr.arpa` is a zone referencing all IP addresses which fall under the `192.168.1.*` IP space.

As one can see, the more specific part of a hostname appears to its left. For example, `example.org.` is more specific than `org.`, as `org.` is more specific than the root zone. The layout of each part of a hostname is much like a file system: the `/dev` directory falls within the root, and so on.

25.6.3. Reasons to Run a Name Server

Name servers usually come in two forms: an authoritative name server, and a caching name server.

An authoritative name server is needed when:

- One wants to serve DNS information to the world, replying authoritatively to queries.
- A domain, such as `example.org`, is registered and IP addresses need to be assigned to hostnames under it.
- An IP address block requires reverse DNS entries (IP to hostname).
- A backup or second name server, called a slave, will reply to queries.

A caching name server is needed when:

- A local DNS server may cache and respond more quickly than querying an outside name server.

When one queries for `www.FreeBSD.org`, the resolver usually queries the uplink ISP's name server, and retrieves the reply. With a local, caching DNS server, the query only has to be made once to the outside world by the caching DNS server. Every additional query will not have to look to the outside of the local network, since the information is cached locally.

25.6.4. How It Works

In FreeBSD, the BIND daemon is called **named** for obvious reasons.

File	Description
<code>named(8)</code>	The BIND daemon.
<code>rndc(8)</code>	Name server control utility.
<code>/etc/namedb</code>	Directory where BIND zone information resides.
<code>/etc/namedb/named.conf</code>	Configuration file of the daemon.

Depending on how a given zone is configured on the server, the files related to that zone can be found in the `master`, `slave`, or `dynamic` subdirectories of the `/etc/namedb` directory. These files contain the DNS information that will be given out by the name server in response to queries.

25.6.5. Starting BIND

Since BIND is installed by default, configuring it all is relatively simple.

The default **named** configuration is that of a basic resolving name server, ran in a `chroot(8)` environment. To start the server one time with this configuration, use the following command:

```
# /etc/rc.d/named forrestart
```

To ensure the **named** daemon is started at boot each time, put the following line into the `/etc/rc.conf`:

```
named_enable="YES"
```

There are obviously many configuration options for `/etc/namedb/named.conf` that are beyond the scope of this document. However, if you are interested in the startup options for **named** on FreeBSD, take a look at the `named_*` flags in `/etc/defaults/rc.conf` and consult the `rc.conf(5)` manual page. The *Sekcja 11.7* section is also a good read.

25.6.6. Configuration Files

Configuration files for **named** currently reside in `/etc/namedb` directory and will need modification before use, unless all that is needed is a simple resolver. This is where most of the configuration will be performed.

25.6.6.1. Using `make-localhost`

To configure a master zone for the localhost visit the `/etc/namedb` directory and run the following command:

```
# sh make-localhost
```

If all went well, a new file should exist in the `master` subdirectory. The filenames should be `localhost.rev` for the local domain name and `localhost-v6.rev` for IPv6 configurations. As the default configuration file, required information will be present in the `named.conf` file.

25.6.6.2. `/etc/namedb/named.conf`

```
// $FreeBSD$
//
// Refer to the named.conf(5) and named(8) man pages, and the documentation
// in /usr/share/doc/bind9 for more details.
//
// If you are going to set up an authoritative server, make sure you
// understand the hairy details of how DNS works. Even with
// simple mistakes, you can break connectivity for affected parties,
// or cause huge amounts of useless Internet traffic.

options {
    directory          "/etc/namedb";
    pid-file            "/var/run/named/pid";
    dump-file           "/var/dump/named_dump.db";
    statistics-file     "/var/stats/named.stats";

    // If named is being used only as a local resolver, this is a safe default.
    // For named to be accessible to the network, comment this option, specify
    // the proper IP address, or delete this option.
    listen-on           { 127.0.0.1; };

    // If you have IPv6 enabled on this system, uncomment this option for
```

```
// use as a local resolver. To give access to the network, specify
// an IPv6 address, or the keyword "any".
//     listen-on-v6    { ::1; };

// In addition to the "forwarders" clause, you can force your name
// server to never initiate queries of its own, but always ask its
// forwarders only, by enabling the following line:
//
//     forward only;

// If you've got a DNS server around at your upstream provider, enter
// its IP address here, and enable the line below. This will make you
// benefit from its cache, thus reduce overall DNS traffic in the Internet.
/*
    forwarders {
        127.0.0.1;
    };
*/
```

Just as the comment says, to benefit from an uplink's cache, forwarders can be enabled here. Under normal circumstances, a name server will recursively query the Internet looking at certain name servers until it finds the answer it is looking for. Having this enabled will have it query the uplink's name server (or name server provided) first, taking advantage of its cache. If the uplink name server in question is a heavily trafficked, fast name server, enabling this may be worthwhile.

Ostrzeżenie 127.0.0.1 will *not* work here. Change this IP address to a name server at your uplink.

```
/*
 * If there is a firewall between you and nameservers you want
 * to talk to, you might need to uncomment the query-source
 * directive below. Previous versions of BIND always asked
 * questions using port 53, but BIND versions 8 and later
 * use a pseudo-random unprivileged UDP port by default.
 */
// query-source address * port 53;
};

// If you enable a local name server, don't forget to enter 127.0.0.1
// first in your /etc/resolv.conf so this server will be queried.
// Also, make sure to enable it in /etc/rc.conf.

zone "." {
    type hint;
    file "named.root";
};

zone "0.0.127.IN-ADDR.ARPA" {
    type master;
    file "master/localhost.rev";
};
```



```
};
zone "1.168.192.in-addr.arpa" {
    type slave;
    file "slave/1.168.192.in-addr.arpa";
    masters {
        192.168.1.1;
    };
};
*/
```

In `named.conf`, these are examples of slave entries for a forward and reverse zone.

For each new zone served, a new zone entry must be added to `named.conf`.

For example, the simplest zone entry for `example.org` can look like:

```
zone "example.org" {
    type master;
    file "master/example.org";
};
```

The zone is a master, as indicated by the `type` statement, holding its zone information in `/etc/namedb/master/example.org` indicated by the `file` statement.

```
zone "example.org" {
    type slave;
    file "slave/example.org";
};
```

In the slave case, the zone information is transferred from the master name server for the particular zone, and saved in the file specified. If and when the master server dies or is unreachable, the slave name server will have the transferred zone information and will be able to serve it.

25.6.6.3. Zone Files

An example master zone file for `example.org` (existing within `/etc/namedb/master/example.org`) is as follows:

```
$TTL 3600          ; 1 hour
example.org.      IN      SOA      ns1.example.org. admin.example.org. (
                                2006051501      ; Serial
                                10800           ; Refresh
                                3600            ; Retry
                                604800          ; Expire
                                86400           ; Minimum TTL
                                )

; DNS Servers
                        IN      NS      ns1.example.org.
                        IN      NS      ns2.example.org.

; MX Records
                        IN      MX 10    mx.example.org.
```

```

                IN      MX 20    mail.example.org.

                IN      A        192.168.1.1

; Machine Names
localhost      IN      A        127.0.0.1
ns1            IN      A        192.168.1.2
ns2            IN      A        192.168.1.3
mx             IN      A        192.168.1.4
mail           IN      A        192.168.1.5

; Aliases
www            IN      CNAME    @

```

Note that every hostname ending in a „.” is an exact hostname, whereas everything without a trailing „.” is referenced to the origin. For example, `www` is translated into `www.origin`. In our fictitious zone file, our origin is `example.org.`, so `www` would translate to `www.example.org.`

The format of a zone file follows:

```
recordname      IN recordtype  value
```

The most commonly used DNS records:

SOA

start of zone authority

NS

an authoritative name server

A

a host address

CNAME

the canonical name for an alias

MX

mail exchanger

PTR

a domain name pointer (used in reverse DNS)

```

example.org. IN SOA ns1.example.org. admin.example.org. (
                        2006051501      ; Serial
                        10800            ; Refresh after 3 hours
                        3600             ; Retry after 1 hour
                        604800           ; Expire after 1 week
                        86400 )          ; Minimum TTL of 1 day

```

`example.org.`

the domain name, also the origin for this zone file.

`ns1.example.org.`

the primary/authoritative name server for this zone.

`admin.example.org.`

the responsible person for this zone, email address with „@” replaced. (<admin@example.org> becomes admin.example.org)

`2006051501`

the serial number of the file. This must be incremented each time the zone file is modified. Nowadays, many admins prefer a `yyyymmddrr` format for the serial number. `2006051501` would mean last modified 05/15/2006, the latter 01 being the first time the zone file has been modified this day. The serial number is important as it alerts slave name servers for a zone when it is updated.

```
IN NS          ns1.example.org.
```

This is an NS entry. Every name server that is going to reply authoritatively for the zone must have one of these entries.

```
localhost      IN      A      127.0.0.1
ns1             IN      A      192.168.1.2
ns2            IN      A      192.168.1.3
mx             IN      A      192.168.1.4
mail           IN      A      192.168.1.5
```

The A record indicates machine names. As seen above, `ns1.example.org` would resolve to `192.168.1.2`.

```
IN      A      192.168.1.1
```

This line assigns IP address `192.168.1.1` to the current origin, in this case `example.org`.

```
www           IN CNAME  @
```

The canonical name record is usually used for giving aliases to a machine. In the example, `www` is aliased to the „master” machine which name equals to domain name `example.org` (`192.168.1.1`). CNAMEs can be used to provide alias hostnames, or round robin one hostname among multiple machines.

```
IN MX  10      mail.example.org.
```

The MX record indicates which mail servers are responsible for handling incoming mail for the zone. `mail.example.org` is the hostname of the mail server, and 10 being the priority of that mail server.

One can have several mail servers, with priorities of 10, 20 and so on. A mail server attempting to deliver to `example.org` would first try the highest priority MX (the record with the lowest priority number), then the second highest, etc, until the mail can be properly delivered.

For `in-addr.arpa` zone files (reverse DNS), the same format is used, except with PTR entries instead of A or CNAME.

`$TTL 3600`

```

1.168.192.in-addr.arpa. IN SOA ns1.example.org. admin.example.org. (
                                2006051501      ; Serial
                                10800            ; Refresh
                                3600             ; Retry
                                604800          ; Expire
                                3600 )          ; Minimum

                                IN      NS      ns1.example.org.
                                IN      NS      ns2.example.org.

1      IN      PTR      example.org.
2      IN      PTR      ns1.example.org.
3      IN      PTR      ns2.example.org.
4      IN      PTR      mx.example.org.
5      IN      PTR      mail.example.org.

```

This file gives the proper IP address to hostname mappings of our above fictitious domain.

25.6.7. Caching Name Server

A caching name server is a name server that is not authoritative for any zones. It simply asks queries of its own, and remembers them for later use. To set one up, just configure the name server as usual, omitting any inclusions of zones.

25.6.8. Security

Although BIND is the most common implementation of DNS, there is always the issue of security. Possible and exploitable security holes are sometimes found.

While FreeBSD automatically drops **named** into a chroot(8) environment; there are several other security mechanisms in place which could help to lure off possible DNS service attacks.

It is always good idea to read CERT (<http://www.cert.org/>)'s security advisories and to subscribe to the Lista dyskusyjna powiadomień bezpieczeństwa FreeBSD

(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications>) to stay up to date with the current Internet and FreeBSD security issues.

Podpowiedź: If a problem arises, keeping sources up to date and having a fresh build of **named** would not hurt.

25.6.9. Further Reading

BIND/**named** manual pages: `rndc(8)` `named(8)` `named.conf(5)`

- Official ISC BIND Page (<http://www.isc.org/products/BIND/>)
- Official ISC BIND Forum (<http://www.isc.org/sw/guild/bf/>)

- BIND FAQ (<http://www.nominum.com/getOpenSourceResource.php?id=6>)
- O'Reilly DNS and BIND 5th Edition (<http://www.oreilly.com/catalog/dns5/>)
- RFC1034 - Domain Names - Concepts and Facilities (<ftp://ftp.isi.edu/in-notes/rfc1034.txt>)
- RFC1035 - Domain Names - Implementation and Specification (<ftp://ftp.isi.edu/in-notes/rfc1035.txt>)

25.7. Apache HTTP Server

Contributed by Murray Stokely.

25.7.1. Overview

FreeBSD is used to run some of the busiest web sites in the world. The majority of web servers on the Internet are using the **Apache HTTP Server**. **Apache** software packages should be included on your FreeBSD installation media. If you did not install **Apache** when you first installed FreeBSD, then you can install it from the [www/apache13](http://www.apache13) or www/apache20 port.

Once **Apache** has been installed successfully, it must be configured.

Notatka: This section covers version 1.3.X of the **Apache HTTP Server** as that is the most widely used version for FreeBSD. **Apache** 2.X introduces many new technologies but they are not discussed here. For more information about **Apache** 2.X, please see <http://httpd.apache.org/>.

25.7.2. Configuration

The main **Apache HTTP Server** configuration file is installed as `/usr/local/etc/apache/httpd.conf` on FreeBSD. This file is a typical UNIX text configuration file with comment lines beginning with the `#` character. A comprehensive description of all possible configuration options is outside the scope of this book, so only the most frequently modified directives will be described here.

```
ServerRoot "/usr/local"
```

This specifies the default directory hierarchy for the **Apache** installation. Binaries are stored in the `bin` and `sbin` subdirectories of the server root, and configuration files are stored in `etc/apache`.

```
ServerAdmin you@your.address
```

The address to which problems with the server should be emailed. This address appears on some server-generated pages, such as error documents.

```
ServerName www.example.com
```

`ServerName` allows you to set a host name which is sent back to clients for your server if it is different to the one that the host is configured with (i.e., use `www` instead of the host's real name).

```
DocumentRoot "/usr/local/www/data"
```

`DocumentRoot`: The directory out of which you will serve your documents. By default, all requests are taken from this directory, but symbolic links and aliases may be used to point to other locations.

It is always a good idea to make backup copies of your **Apache** configuration file before making changes. Once you are satisfied with your initial configuration you are ready to start running **Apache**.

25.7.3. Running Apache

Apache does not run from the **inetd** super server as many other network servers do. It is configured to run standalone for better performance for incoming HTTP requests from client web browsers. A shell script wrapper is included to make starting, stopping, and restarting the server as simple as possible. To start up **Apache** for the first time, just run:

```
# /usr/local/sbin/apachectl start
```

You can stop the server at any time by typing:

```
# /usr/local/sbin/apachectl stop
```

After making changes to the configuration file for any reason, you will need to restart the server:

```
# /usr/local/sbin/apachectl restart
```

To restart **Apache** without aborting current connections, run:

```
# /usr/local/sbin/apachectl graceful
```

Additional information available at `apachectl(8)` manual page.

To launch **Apache** at system startup, add the following line to `/etc/rc.conf`:

```
apache_enable="YES"
```

If you would like to supply additional command line options for the **Apache** `httpd` program started at system boot, you may specify them with an additional line in `rc.conf`:

```
apache_flags=" "
```

Now that the web server is running, you can view your web site by pointing a web browser to `http://localhost/`. The default web page that is displayed is `/usr/local/www/data/index.html`.

25.7.4. Virtual Hosting

Apache supports two different types of Virtual Hosting. The first method is Name-based Virtual Hosting. Name-based virtual hosting uses the clients HTTP/1.1 headers to figure out the hostname. This allows many different domains to share the same IP address.

To setup **Apache** to use Name-based Virtual Hosting add an entry like the following to your `httpd.conf`:

```
NameVirtualHost *
```

If your webserver was named `www.domain.tld` and you wanted to setup a virtual domain for `www.someotherdomain.tld` then you would add the following entries to `httpd.conf`:

```
<VirtualHost *>
ServerName www.domain.tld
DocumentRoot /www/domain.tld
</VirtualHost>

<VirtualHost *>
ServerName www.someotherdomain.tld
DocumentRoot /www/someotherdomain.tld
</VirtualHost>
```

Replace the addresses with the addresses you want to use and the path to the documents with what you are using.

For more information about setting up virtual hosts, please consult the official **Apache** documentation at: <http://httpd.apache.org/docs/vhosts/>.

25.7.5. Apache Modules

There are many different **Apache** modules available to add functionality to the basic server. The FreeBSD Ports Collection provides an easy way to install **Apache** together with some of the more popular add-on modules.

25.7.5.1. mod_ssl

The **mod_ssl** module uses the OpenSSL library to provide strong cryptography via the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols. This module provides everything necessary to request a signed certificate from a trusted certificate signing authority so that you can run a secure web server on FreeBSD.

If you have not yet installed **Apache**, then a version of **Apache** 1.3.X that includes **mod_ssl** may be installed with the `www/apache13-modssl` port. SSL support is also available for **Apache** 2.X in the `www/apache20` port, where it is enabled by default.

25.7.5.2. Dynamic Websites with Perl & PHP

In the past few years, more businesses have turned to the Internet in order to enhance their revenue and increase exposure. This has also increased the need for interactive web content. While some companies, such as Microsoft, have introduced solutions into their proprietary products, the open source community answered the call. Two options for dynamic web content include **mod_perl** & **mod_php**.

25.7.5.2.1. mod_perl

The **Apache/Perl** integration project brings together the full power of the Perl programming language and the **Apache HTTP Server**. With the **mod_perl** module it is possible to write **Apache** modules entirely in Perl. In addition, the persistent interpreter embedded in the server avoids the overhead of starting an external interpreter and the penalty of Perl start-up time.

mod_perl is available a few different ways. To use **mod_perl** remember that **mod_perl** 1.0 only works with **Apache** 1.3 and **mod_perl** 2.0 only works with **Apache** 2. **mod_perl** 1.0 is available in `www/mod_perl` and a statically compiled version is available in `www/apache13-modperl`. **mod_perl** 2.0 is available in `www/mod_perl2`.

25.7.5.2.2. *mod_php*

Written by Tom Rhodes.

PHP, also known as „PHP: Hypertext Preprocessor” is a general-purpose scripting language that is especially suited for Web development. Capable of being embedded into HTML its syntax draws upon C, Java, and Perl with the intention of allowing web developers to write dynamically generated webpages quickly.

To gain support for PHP5 for the **Apache** web server, begin by installing the `www/mod_php5` port.

This will install and configure the modules required to support dynamic PHP applications. Check to ensure the following sections have been added to `/usr/local/etc/apache/httpd.conf`:

```
LoadModule php5_module          libexec/apache/libphp5.so

AddModule mod_php5.c
    <IfModule mod_php5.c>
        DirectoryIndex index.php index.html
    </IfModule>
    <IfModule mod_php5.c>
        AddType application/x-httpd-php .php
        AddType application/x-httpd-php-source .phps
    </IfModule>
```

Once completed, a simple call to the `apachectl` command for a graceful restart is needed to load the PHP module:

```
# apachectl graceful
```

The PHP support in FreeBSD is extremely modular so the base install is very limited. It is very easy to add support using the `lang/php5-extensions` port. This port provides a menu driven interface to PHP extension installation. Alternatively, individual extensions can be installed using the appropriate port.

For instance, to add support for the **MySQL** database server to PHP5, simply install the `databases/php5-mysql` port.

After installing an extension, the **Apache** server must be reloaded to pick up the new configuration changes:

```
# apachectl graceful
```

25.8. File Transfer Protocol (FTP)

Contributed by Murray Stokely.

25.8.1. Overview

The File Transfer Protocol (FTP) provides users with a simple way to transfer files to and from an FTP server. FreeBSD includes FTP server software, **ftpd**, in the base system. This makes setting up and administering an FTP server on FreeBSD very straightforward.

25.8.2. Configuration

The most important configuration step is deciding which accounts will be allowed access to the FTP server. A normal FreeBSD system has a number of system accounts used for various daemons, but unknown users should not be allowed to log in with these accounts. The `/etc/ftpusers` file is a list of users disallowed any FTP access. By default, it includes the aforementioned system accounts, but it is possible to add specific users here that should not be allowed access to FTP.

You may want to restrict the access of some users without preventing them completely from using FTP. This can be accomplished with the `/etc/ftpchroot` file. This file lists users and groups subject to FTP access restrictions. The `ftpchroot(5)` manual page has all of the details so it will not be described in detail here.

If you would like to enable anonymous FTP access to your server, then you must create a user named `ftp` on your FreeBSD system. Users will then be able to log on to your FTP server with a username of `ftp` or `anonymous` and with any password (by convention an email address for the user should be used as the password). The FTP server will call `chroot(2)` when an anonymous user logs in, to restrict access to only the home directory of the `ftp` user.

There are two text files that specify welcome messages to be displayed to FTP clients. The contents of the file `/etc/ftpwelcome` will be displayed to users before they reach the login prompt. After a successful login, the contents of the file `/etc/ftpmotd` will be displayed. Note that the path to this file is relative to the login environment, so the file `~ftp/etc/ftpmotd` would be displayed for anonymous users.

Once the FTP server has been configured properly, it must be enabled in `/etc/inetd.conf`. All that is required here is to remove the comment symbol „#” from in front of the existing **ftpd** line :

```
ftp      stream  tcp      nowait  root    /usr/libexec/ftpd      ftpd -l
```

As explained in Przykład 25-1, the **inetd** configuration must be reloaded after this configuration file is changed.

You can now log on to your FTP server by typing:

```
% ftp localhost
```

25.8.3. Maintaining

The **ftpd** daemon uses `syslog(3)` to log messages. By default, the system log daemon will put messages related to FTP in the `/var/log/xferlog` file. The location of the FTP log can be modified by changing the following line in `/etc/syslog.conf`:

```
ftp.info      /var/log/xferlog
```

Be aware of the potential problems involved with running an anonymous FTP server. In particular, you should think twice about allowing anonymous users to upload files. You may find that your FTP site becomes a forum for the trade of unlicensed commercial software or worse. If you do need to allow anonymous FTP uploads, then you should set up the permissions so that these files can not be read by other anonymous users until they have been reviewed.

25.9. File and Print Services for Microsoft® Windows clients (Samba)

Contributed by Murray Stokely.

25.9.1. Overview

Samba is a popular open source software package that provides file and print services for Microsoft Windows clients. Such clients can connect to and use FreeBSD filesystems as if it was a local disk drive, or FreeBSD printers as if they were local printers.

Samba software packages should be included on your FreeBSD installation media. If you did not install **Samba** when you first installed FreeBSD, then you can install it from the `net/samba3` port or package.

25.9.2. Configuration

A default **Samba** configuration file is installed as `/usr/local/etc/smb.conf.default`. This file must be copied to `/usr/local/etc/smb.conf` and customized before **Samba** can be used.

The `smb.conf` file contains runtime configuration information for **Samba**, such as definitions of the printers and „file system shares” that you would like to share with Windows clients. The **Samba** package includes a web based tool called **swat** which provides a simple way of configuring the `smb.conf` file.

25.9.2.1. Using the Samba Web Administration Tool (SWAT)

The Samba Web Administration Tool (SWAT) runs as a daemon from **inetd**. Therefore, the following line in `/etc/inetd.conf` should be uncommented before **swat** can be used to configure **Samba**:

```
swat    stream  tcp      nowait/400    root    /usr/local/sbin/swat
```

As explained in Przykład 25-1, the **inetd** must be reloaded after this configuration file is changed.

Once **swat** has been enabled in `inetd.conf`, you can use a browser to connect to `http://localhost:901`. You will first have to log on with the system `root` account.

Once you have successfully logged on to the main **Samba** configuration page, you can browse the system documentation, or begin by clicking on the **Globals** tab. The **Globals** section corresponds to the variables that are set in the `[global]` section of `/usr/local/etc/smb.conf`.

25.9.2.2. Global Settings

Whether you are using **swat** or editing `/usr/local/etc/smb.conf` directly, the first directives you are likely to encounter when configuring **Samba** are:

```
workgroup
```

NT Domain-Name or Workgroup-Name for the computers that will be accessing this server.

```
netbios name
```

This sets the NetBIOS name by which a **Samba** server is known. By default it is the same as the first component of the host's DNS name.

```
server string
```

This sets the string that will be displayed with the `net view` command and some other networking tools that seek to display descriptive text about the server.

25.9.2.3. Security Settings

Two of the most important settings in `/usr/local/etc/smb.conf` are the security model chosen, and the backend password format for client users. The following directives control these options:

```
security
```

The two most common options here are `security = share` and `security = user`. If your clients use usernames that are the same as their usernames on your FreeBSD machine then you will want to use user level security. This is the default security policy and it requires clients to first log on before they can access shared resources.

In share level security, client do not need to log onto the server with a valid username and password before attempting to connect to a shared resource. This was the default security model for older versions of **Samba**.

```
passwd backend
```

Samba has several different backend authentication models. You can authenticate clients with LDAP, NIS+, a SQL database, or a modified password file. The default authentication method is `smbpasswd`, and that is all that will be covered here.

Assuming that the default `smbpasswd` backend is used, the `/usr/local/private/smbpasswd` file must be created to allow **Samba** to authenticate clients. If you would like to give all of your UNIX user accounts access from Windows clients, use the following command:

```
# grep -v "^#" /etc/passwd | make_smbpasswd > /usr/local/private/smbpasswd
# chmod 600 /usr/local/private/smbpasswd
```

Please see the **Samba** documentation for additional information about configuration options. With the basics outlined here, you should have everything you need to start running **Samba**.

25.9.3. Starting Samba

To enable **Samba** when your system boots, add the following line to `/etc/rc.conf`:

```
samba_enable="YES"
```

You can then start **Samba** at any time by typing:

```
# /usr/local/etc/rc.d/samba.sh start
Starting SAMBA: removing stale tdb's :
Starting nmbd.
Starting smbd.
```

Samba actually consists of three separate daemons. You should see that both the **nmbd** and **smbd** daemons are started by the `samba.sh` script. If you enabled winbind name resolution services in `smb.conf`, then you will also see that the **winbindd** daemon is started.

You can stop **Samba** at any time by typing :

```
# /usr/local/etc/rc.d/samba.sh stop
```

Samba is a complex software suite with functionality that allows broad integration with Microsoft Windows networks. For more information about functionality beyond the basic installation described here, please see <http://www.samba.org>.

25.10. Clock Synchronization with NTP

Contributed by Tom Hukins.

25.10.1. Overview

Over time, a computer's clock is prone to drift. The Network Time Protocol (NTP) is one way to ensure your clock stays accurate.

Many Internet services rely on, or greatly benefit from, computers' clocks being accurate. For example, a web server may receive requests to send a file if it has been modified since a certain time. In a local area network environment, it is essential that computers sharing files from the same file server have synchronized clocks so that file timestamps stay consistent. Services such as `cron(8)` also rely on an accurate system clock to run commands at the specified times.

FreeBSD ships with the `ntpd(8)` NTP server which can be used to query other NTP servers to set the clock on your machine or provide time services to others.

25.10.2. Choosing Appropriate NTP Servers

In order to synchronize your clock, you will need to find one or more NTP servers to use. Your network administrator or ISP may have set up an NTP server for this purpose—check their documentation to see if this is the case. There is an online list of publicly accessible NTP servers (<http://ntp.isc.org/bin/view/Servers/WebHome>) which you can use to find an NTP server near to you. Make sure you are aware of the policy for any servers you choose, and ask for permission if required.

Choosing several unconnected NTP servers is a good idea in case one of the servers you are using becomes unreachable or its clock is unreliable. `ntpd(8)` uses the responses it receives from other servers intelligently—it will favor unreliable servers less than reliable ones.

25.10.3. Configuring Your Machine

25.10.3.1. Basic Configuration

If you only wish to synchronize your clock when the machine boots up, you can use `ntpdate(8)`. This may be

appropriate for some desktop machines which are frequently rebooted and only require infrequent synchronization, but most machines should run `ntpd(8)`.

Using `ntpdate(8)` at boot time is also a good idea for machines that run `ntpd(8)`. The `ntpd(8)` program changes the clock gradually, whereas `ntpdate(8)` sets the clock, no matter how great the difference between a machine's current clock setting and the correct time.

To enable `ntpdate(8)` at boot time, add `ntpdate_enable="YES"` to `/etc/rc.conf`. You will also need to specify all servers you wish to synchronize with and any flags to be passed to `ntpdate(8)` in `ntpdate_flags`.

25.10.3.2. General Configuration

NTP is configured by the `/etc/ntp.conf` file in the format described in `ntp.conf(5)`. Here is a simple example:

```
server ntplocal.example.com prefer
server timeserver.example.org
server ntp2a.example.net
```

```
driftfile /var/db/ntp.drift
```

The `server` option specifies which servers are to be used, with one server listed on each line. If a server is specified with the `prefer` argument, as with `ntplocal.example.com`, that server is preferred over other servers. A response from a preferred server will be discarded if it differs significantly from other servers' responses, otherwise it will be used without any consideration to other responses. The `prefer` argument is normally used for NTP servers that are known to be highly accurate, such as those with special time monitoring hardware.

The `driftfile` option specifies which file is used to store the system clock's frequency offset. The `ntpd(8)` program uses this to automatically compensate for the clock's natural drift, allowing it to maintain a reasonably correct setting even if it is cut off from all external time sources for a period of time.

The `driftfile` option specifies which file is used to store information about previous responses from the NTP servers you are using. This file contains internal information for NTP. It should not be modified by any other process.

25.10.3.3. Controlling Access to Your Server

By default, your NTP server will be accessible to all hosts on the Internet. The `restrict` option in `/etc/ntp.conf` allows you to control which machines can access your server.

If you want to deny all machines from accessing your NTP server, add the following line to `/etc/ntp.conf`:

```
restrict default ignore
```

If you only want to allow machines within your own network to synchronize their clocks with your server, but ensure they are not allowed to configure the server or used as peers to synchronize against, add

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

instead, where `192.168.1.0` is an IP address on your network and `255.255.255.0` is your network's netmask.

`/etc/ntp.conf` can contain multiple `restrict` options. For more details, see the `Access Control Support` subsection of `ntp.conf(5)`.

25.10.4. Running the NTP Server

To ensure the NTP server is started at boot time, add the line `ntpd_enable="YES"` to `/etc/rc.conf`. If you wish to pass additional flags to `ntpd(8)`, edit the `ntpd_flags` parameter in `/etc/rc.conf`.

To start the server without rebooting your machine, run `ntpd` being sure to specify any additional parameters from `ntpd_flags` in `/etc/rc.conf`. For example:

```
# ntpd -p /var/run/ntpd.pid
```

25.10.5. Using ntpd with a Temporary Internet Connection

The `ntpd(8)` program does not need a permanent connection to the Internet to function properly. However, if you have a temporary connection that is configured to dial out on demand, it is a good idea to prevent NTP traffic from triggering a dial out or keeping the connection alive. If you are using user PPP, you can use `filter` directives in `/etc/ppp/ppp.conf`. For example:

```
set filter dial 0 deny udp src eq 123
# Prevent NTP traffic from initiating dial out
set filter dial 1 permit 0 0
set filter alive 0 deny udp src eq 123
# Prevent incoming NTP traffic from keeping the connection open
set filter alive 1 deny udp dst eq 123
# Prevent outgoing NTP traffic from keeping the connection open
set filter alive 2 permit 0/0 0/0
```

For more details see the `PACKET FILTERING` section in `ppp(8)` and the examples in `/usr/share/examples/ppp/`.

Notatka: Some Internet access providers block low-numbered ports, preventing NTP from functioning since replies never reach your machine.

25.10.6. Further Information

Documentation for the NTP server can be found in `/usr/share/doc/ntp/` in HTML format.

Rozdział 26. Firewalls

Contributed by Joseph J. Barbish. Converted to SGML and updated by Brad Davis.

26.1. Introduction

Firewalls make it possible to filter incoming and outgoing traffic that flows through your system. A firewall can use one or more sets of „rules” to inspect the network packets as they come in or go out of your network connections and either allows the traffic through or blocks it. The rules of a firewall can inspect one or more characteristics of the packets, including but not limited to the protocol type, the source or destination host address, and the source or destination port.

Firewalls can greatly enhance the security of a host or a network. They can be used to do one or more of the following things:

- To protect and insulate the applications, services and machines of your internal network from unwanted traffic coming in from the public Internet.
- To limit or disable access from hosts of the internal network to services of the public Internet.
- To support network address translation (NAT), which allows your internal network to use private IP addresses and share a single connection to the public Internet (either with a single IP address or by a shared pool of automatically assigned public addresses).

After reading this chapter, you will know:

- How to properly define packet filtering rules.
- The differences between the firewalls built into FreeBSD.
- How to use and configure the OpenBSD **PF** firewall.
- How to use and configure **IPFILTER**.
- How to use and configure **IPFW**.

Before reading this chapter, you should:

- Understand basic FreeBSD and Internet concepts.

26.2. Firewall Concepts

There are two basic ways to create firewall rulesets: „inclusive” or „exclusive”. An exclusive firewall allows all traffic through except for the traffic matching the ruleset. An inclusive firewall does the reverse. It only allows traffic matching the rules through and blocks everything else.

Inclusive firewalls are generally safer than exclusive firewalls because they significantly reduce the risk of allowing unwanted traffic to pass through the firewall.

Security can be tightened further using a „stateful firewall”. With a stateful firewall the firewall keeps track of which connections are opened through the firewall and will only allow traffic through which either matches an existing connection or opens a new one. The disadvantage of a stateful firewall is that it can be vulnerable to Denial of

Service (DoS) attacks if a lot of new connections are opened very fast. With most firewalls it is possible to use a combination of stateful and non-stateful behavior to make an optimal firewall for the site.

26.3. Firewall Packages

FreeBSD has three different firewall packages built into the base system. They are: *IPFILTER* (also known as IPF), *IPFIREWALL* (also known as IPFW), and *OpenBSD's PacketFilter* (also known as PF). FreeBSD also has two built in packages for traffic shaping (basically controlling bandwidth usage): *altq(4)* and *dummynet(4)*. Dummynet has traditionally been closely tied with IPFW, and ALTQ with IPF/PF. IPF, IPFW, and PF all use rules to control the access of packets to and from your system, although they go about it different ways and have different rule syntaxes.

The reason that FreeBSD has multiple built in firewall packages is that different people have different requirements and preferences. No single firewall package is the best.

The author prefers IPFILTER because its stateful rules are much less complicated to use in a NAT environment and it has a built in ftp proxy that simplifies the rules to allow secure outbound FTP usage.

Since all firewalls are based on inspecting the values of selected packet control fields, the creator of the firewall rulesets must have an understanding of how TCP/IP works, what the different values in the packet control fields are and how these values are used in a normal session conversation. For a good explanation go to: <http://www.ipprimer.com/overview.cfm>.

26.4. The OpenBSD Packet Filter (PF) and ALTQ

As of July 2003 the OpenBSD firewall software application known as PF was ported to FreeBSD and was made available in the FreeBSD Ports Collection; the first release that contained PF as an integrated part of the base system was FreeBSD 5.3 in November 2004. PF is a complete, fully featured firewall that has optional support for ALTQ (Alternate Queuing). ALTQ provides Quality of Service (QoS) bandwidth shaping that allows guaranteeing bandwidth to different services based on filtering rules. The OpenBSD Project does an outstanding job of maintaining the PF User's Guide that it will not be made part of this handbook firewall section as that would just be duplicated effort.

More info can be found at the PF for FreeBSD web site: <http://pf4freebsd.love2party.net/>.

26.4.1. Enabling PF

PF is included in the basic FreeBSD install for versions newer than 5.3 as a separate run time loadable module. The system will dynamically load the PF kernel loadable module when the `rc.conf` statement `pf_enable="YES"` is used. The loadable module was created with `pflog(4)` logging enabled.

Notatka: The module assumes the presence of `options INET` and `device bpf`. Unless `NOINET6` for FreeBSD prior to 6.0-RELEASE and `NO_INET6` for later releases (for example in `make.conf(5)`) was defined during the build, it also requires `options INET6`.

Once the kernel module is loaded or the kernel is statically built with PF support, it is possible to enable or disable **pf** with the `pfctl` command.

This example demonstrates how to enable **pf**:

```
# pfctl -e
```

The `pfctl` command provides a way to work with the **pf** firewall. It is a good idea to check the `pfctl(8)` manual page to find out more information about using it.

26.4.2. Kernel options

It is not a mandatory requirement that you enable PF by compiling the following options into the FreeBSD kernel. It is only presented here as background information. Compiling PF into the kernel causes the loadable module to never be used.

Sample kernel config PF option statements are in the `/usr/src/sys/conf/NOTES` kernel source and are reproduced here:

```
device pf
device pflog
device pfsync
```

`device pf` enables support for the „Packet Filter” firewall.

`device pflog` enables the optional `pflog(4)` pseudo network device which can be used to log traffic to a `bpf(4)` descriptor. The `pflogd(8)` daemon can be used to store the logging information to disk.

`device pfsync` enables the optional `pfsync(4)` pseudo network device that is used to monitor „state changes”. As this is not part of the loadable module one has to build a custom kernel to use it.

These settings will take effect only after you have built and installed a kernel with them set.

26.4.3. Available rc.conf Options

You need the following statements in `/etc/rc.conf` to activate PF at boot time:

```
pf_enable="YES"                # Enable PF (load module if required)
pf_rules="/etc/pf.conf"        # rules definition file for pf
pf_flags=""                    # additional flags for pfctl startup
pflog_enable="YES"             # start pflogd(8)
pflog_logfile="/var/log/pflog" # where pflogd should store the logfile
pflog_flags=""                 # additional flags for pflogd startup
```

If you have a LAN behind this firewall and have to forward packets for the computers in the LAN or want to do NAT, you have to enable the following option as well:

```
gateway_enable="YES"           # Enable as LAN gateway
```

26.4.4. Enabling ALTQ

ALTQ is only available by compiling the options into the FreeBSD Kernel. ALTQ is not supported by all of the available network card drivers. Please see the `altq(4)` manual page for a list of drivers that are supported in your release of FreeBSD. The following options will enable ALTQ and add additional functionality.

```
options          ALTQ
```

```

options      ALTQ_CBQ      # Class Bases Queuing (CBQ)
options      ALTQ_RED      # Random Early Detection (RED)
options      ALTQ_RIO      # RED In/Out
options      ALTQ_HFSC     # Hierarchical Packet Scheduler (HFSC)
options      ALTQ_PRIQ     # Priority Queuing (PRIQ)
options      ALTQ_NOPCC    # Required for SMP build

```

`options ALTQ` enables the ALTQ framework.

`options ALTQ_CBQ` enables Class Based Queuing (CBQ). CBQ allows you to divide a connection's bandwidth into different classes or queues to prioritize traffic based on filter rules.

`options ALTQ_RED` enables Random Early Detection (RED). RED is used to avoid network congestion. RED does this by measuring the length of the queue and comparing it to the minimum and maximum thresholds for the queue. If the queue is over the maximum all new packets will be dropped. True to its name, RED drops packets from different connections randomly.

`options ALTQ_RIO` enables Random Early Detection In and Out.

`options ALTQ_HFSC` enables the Hierarchical Fair Service Curve Packet Scheduler. For more information about HFSC see: <http://www-2.cs.cmu.edu/~h Zhang/HFSC/main.html>.

`options ALTQ_PRIQ` enables Priority Queuing (PRIQ). PRIQ will always pass traffic that is in a higher queue first.

`options ALTQ_NOPCC` enables SMP support for ALTQ. This option is required on SMP systems.

26.4.5. Creating Filtering Rules

The Packet Filter reads its configuration rules from the `pf.conf(5)` file and it modifies, drops or passes packets according to the rules or definitions specified there. The FreeBSD installation comes with a default `/etc/pf.conf` which contains useful examples and explanations.

Although FreeBSD has its own `/etc/pf.conf` the syntax is the same as one used in OpenBSD. A great resource for configuring the **pf** firewall has been written by OpenBSD team and is available at <http://www.openbsd.org/faq/pf/>.

Ostrzeżenie When browsing the pf user's guide, please keep in mind that different versions of FreeBSD contain different versions of pf. The **pf** firewall in FreeBSD 5.X is at the level of OpenBSD version 3.5 and in FreeBSD 6.X is at the level of OpenBSD version 3.7.

The Lista dyskusyjna zapory Packet Filter we FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf>) is a good place to ask questions about configuring and running the **pf** firewall. Do not forget to check the mailing list archives before asking questions.

26.5. The IPFILTER (IPF) Firewall

Notatka: This section is work in progress. The contents might not be accurate at all times.

The author of IPFILTER is Darren Reed. IPFILTER is not operating system dependent: it is an open source application and has been ported to FreeBSD, NetBSD, OpenBSD, SunOS, HP/UX, and Solaris operating systems. IPFILTER is actively being supported and maintained, with updated versions being released regularly.

IPFILTER is based on a kernel-side firewall and NAT mechanism that can be controlled and monitored by userland interface programs. The firewall rules can be set or deleted with the `ipf(8)` utility. The NAT rules can be set or deleted with the `ipnat(1)` utility. The `ipfstat(8)` utility can print run-time statistics for the kernel parts of IPFILTER. The `ipmon(8)` program can log IPFILTER actions to the system log files.

IPF was originally written using a rule processing logic of „the last matching rule wins” and used only stateless type of rules. Over time IPF has been enhanced to include a „quick” option and a stateful „keep state” option which drastically modernized the rules processing logic. IPF’s official documentation covers the legacy rule coding parameters and the legacy rule file processing logic. The modernized functions are only included as additional options, completely understating their benefits in producing a far superior secure firewall.

The instructions contained in this section are based on using rules that contain the „quick” option and the stateful „keep state” option. This is the basic framework for coding an inclusive firewall rule set.

An inclusive firewall only allows packets matching the rules to pass through. This way you can control what services can originate behind the firewall destined for the public Internet and also control the services which can originate from the public Internet accessing your private network. Everything else is blocked and logged by default design. Inclusive firewalls are much, much more secure than exclusive firewall rule sets and is the only rule set type covered herein.

For detailed explanation of the legacy rules processing method see:

http://www.obfuscation.org/ipf/ipf-howto.html#TOC_1 and <http://coombs.anu.edu.au/~avalon/ip-filter.html>.

The IPF FAQ is at <http://www.phildev.net/ipf/index.html>.

A searchable archive of the open-source IPFilter mailing list is available at <http://marc.theaimsgroup.com/?l=ipfilter>.

26.5.1. Enabling IPF

IPF is included in the basic FreeBSD install as a separate run time loadable module. The system will dynamically load the IPF kernel loadable module when the `rc.conf` statement `ipfilter_enable="YES"` is used. The loadable module was created with logging enabled and the `default pass all` options. You do not need to compile IPF into the FreeBSD kernel just to change the default to `block all`, you can do that by just coding a `block all` rule at the end of your rule set.

26.5.2. Kernel options

It is not a mandatory requirement that you enable IPF by compiling the following options into the FreeBSD kernel. It is only presented here as background information. Compiling IPF into the kernel causes the loadable module to never be used.

Sample kernel config IPF option statements are in the `/usr/src/sys/conf/NOTES` kernel source and are reproduced here:

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_DEFAULT_BLOCK
```

`options IPFILTER` enables support for the „IPFILTER” firewall.

`options IPFILTER_LOG` enables the option to have IPF log traffic by writing to the `ipl` packet logging pseudo—device for every rule that has the `log` keyword.

`options IPFILTER_DEFAULT_BLOCK` changes the default behavior so any packet not matching a firewall pass rule gets blocked.

These settings will take effect only after you have built and installed a kernel with them set.

26.5.3. Available rc.conf Options

You need the following statements in `/etc/rc.conf` to activate IPF at boot time:

```
ipfilter_enable="YES"           # Start ipf firewall
ipfilter_rules="/etc/ipf.rules" # loads rules definition text file
ipmon_enable="YES"             # Start IP monitor log
ipmon_flags="-Ds"              # D = start as daemon
                                # s = log to syslog
                                # v = log tcp window, ack, seq
                                # n = map IP & port to names
```

If you have a LAN behind this firewall that uses the reserved private IP address ranges, then you need to add the following to enable NAT functionality:

```
gateway_enable="YES"           # Enable as LAN gateway
ipnat_enable="YES"             # Start ipnat function
ipnat_rules="/etc/ipnat.rules" # rules definition file for ipnat
```

26.5.4. IPF

The `ipf` command is used to load your rules file. Normally you create a file containing your custom rules and use this command to replace in mass the currently running firewall internal rules:

```
# ipf -Fa -f /etc/ipf.rules
```

`-Fa` means flush all internal rules tables.

`-f` means this is the file to read for the rules to load.

This gives you the ability to make changes to your custom rules file, run the above IPF command, and thus update the running firewall with a fresh copy of all the rules without having to reboot the system. This method is very convenient for testing new rules as the procedure can be executed as many times as needed.

See the `ipf(8)` manual page for details on the other flags available with this command.

The `ipf(8)` command expects the rules file to be a standard text file. It will not accept a rules file written as a script with symbolic substitution.

There is a way to build IPF rules that utilizes the power of script symbolic substitution. For more information, see Sekcja 26.5.9.

26.5.5. IPFSTAT

The default behavior of `ipfstat(8)` is to retrieve and display the totals of the accumulated statistics gathered as a result of applying the user coded rules against packets going in and out of the firewall since it was last started, or since the last time the accumulators were reset to zero by the `ipf -Z` command.

See the `ipfstat(8)` manual page for details.

The default `ipfstat(8)` command output will look something like this:

```
input packets: blocked 99286 passed 1255609 nomatch 14686 counted 0
output packets: blocked 4200 passed 1284345 nomatch 14687 counted 0
input packets logged: blocked 99286 passed 0
output packets logged: blocked 0 passed 0
packets logged: input 0 output 0
log failures: input 3898 output 0
fragment state(in): kept 0 lost 0
fragment state(out): kept 0 lost 0
packet state(in): kept 169364 lost 0
packet state(out): kept 431395 lost 0
ICMP replies: 0 TCP RSTs sent: 0
Result cache hits(in): 1215208 (out): 1098963
IN Pullups succeeded: 2 failed: 0
OUT Pullups succeeded: 0 failed: 0
Fastroute successes: 0 failures: 0
TCP cksum fails(in): 0 (out): 0
Packet log flags set: (0)
```

When supplied with either `-i` for inbound or `-o` for outbound, it will retrieve and display the appropriate list of filter rules currently installed and in use by the kernel.

`ipfstat -in` displays the inbound internal rules table with rule number.

`ipfstat -on` displays the outbound internal rules table with the rule number.

The output will look something like this:

```
@1 pass out on xl0 from any to any
@2 block out on dc0 from any to any
@3 pass out quick on dc0 proto tcp/udp from any to any keep state
```

`ipfstat -ih` displays the inbound internal rules table, prefixing each rule with a count of how many times the rule was matched.

`ipfstat -oh` displays the outbound internal rules table, prefixing each rule with a count of how many times the rule was matched.

The output will look something like this:

```
2451423 pass out on xl0 from any to any
354727 block out on dc0 from any to any
430918 pass out quick on dc0 proto tcp/udp from any to any keep state
```

One of the most important functions of the `ipfstat` command is the `-t` flag which displays the state table in a way similar to the way `top(1)` shows the FreeBSD running process table. When your firewall is under attack this function gives you the ability to identify, drill down to, and see the attacking packets. The optional sub-flags give the ability to

select the destination or source IP, port, or protocol that you want to monitor in real time. See the `ipfstat(8)` manual page for details.

26.5.6. IPMON

In order for `ipmon` to work properly, the kernel option `IPFILTER_LOG` must be turned on. This command has two different modes that it can be used in. Native mode is the default mode when you type the command on the command line without the `-D` flag.

Daemon mode is for when you want to have a continuous system log file available so that you can review logging of past events. This is how FreeBSD and `IPFILTER` are configured to work together. FreeBSD has a built in facility to automatically rotate system logs. That is why outputting the log information to `syslogd` is better than the default of outputting to a regular file. In the default `rc.conf` file you see the `ipmon_flags` statement uses the `-Ds` flags:

```
ipmon_flags="-Ds" # D = start as daemon
                  # s = log to syslog
                  # v = log tcp window, ack, seq
                  # n = map IP & port to names
```

The benefits of logging are obvious. It provides the ability to review, after the fact, information such as which packets had been dropped, what addresses they came from and where they were going. These all give you a significant edge in tracking down attackers.

Even with the logging facility enabled, `IPF` will not generate any rule logging on its own. The firewall administrator decides what rules in the rule set he wants to log and adds the `log` keyword to those rules. Normally only deny rules are logged.

It is very customary to include a default deny everything rule with the `log` keyword included as your last rule in the rule set. This way you get to see all the packets that did not match any of the rules in the rule set.

26.5.7. IPMON Logging

Syslogd uses its own special method for segregation of log data. It uses special groupings called „facility” and „level”. `IPMON` in `-Ds` mode uses `security` as the „facility” name. All `IPMON` logged data goes to `security`. The following levels can be used to further segregate the logged data if desired:

```
LOG_INFO - packets logged using the "log" keyword as the action rather than pass or block.
LOG_NOTICE - packets logged which are also passed
LOG_WARNING - packets logged which are also blocked
LOG_ERR - packets which have been logged and which can be considered short
```

To setup `IPFILTER` to log all data to `/var/log/ipfilter.log`, you will need to create the file. The following command will do that:

```
# touch /var/log/ipfilter.log
```

The `syslog` function is controlled by definition statements in the `/etc/syslog.conf` file. The `syslog.conf` file offers considerable flexibility in how `syslog` will deal with system messages issued by software applications like `IPF`.

Add the following statement to `/etc/syslog.conf`:

```
security.* /var/log/ipfilter.log
```

Or add the following statement to `/etc/syslog.conf`.

The `security.*` means to write all the logged messages to the coded file location.

To activate the changes to `/etc/syslog.conf` you can reboot or bump the syslog task into re-reading `/etc/syslog.conf` by running `/etc/rc.d/syslogd reload`

Do not forget to change `/etc/newsyslog.conf` to rotate the new log you just created above.

26.5.8. The Format of Logged Messages

Messages generated by `ipmon` consist of data fields separated by white space. Fields common to all messages are:

1. The date of packet receipt.
2. The time of packet receipt. This is in the form `HH:MM:SS.F`, for hours, minutes, seconds, and fractions of a second (which can be several digits long).
3. The name of the interface the packet was processed on, e.g. `dc0`.
4. The group and rule number of the rule, e.g. `@0:17`.

These can be viewed with `ipfstat -in`.

1. The action: `p` for passed, `b` for blocked, `S` for a short packet, `n` did not match any rules, `L` for a log rule. The order of precedence in showing flags is: `S`, `p`, `b`, `n`, `L`. A capital `P` or `B` means that the packet has been logged due to a global logging setting, not a particular rule.
2. The addresses. This is actually three fields: the source address and port (separated by a comma), the `->` symbol, and the destination address and port. `209.53.17.22,80->198.73.220.17,1722`.
3. `PR` followed by the protocol name or number, e.g. `PR tcp`.
4. `len` followed by the header length and total length of the packet, e.g. `len 20 40`.

If the packet is a TCP packet, there will be an additional field starting with a hyphen followed by letters corresponding to any flags that were set. See the `ipmon(8)` manual page for a list of letters and their flags.

If the packet is an ICMP packet, there will be two fields at the end, the first always being „ICMP”, and the next being the ICMP message and sub-message type, separated by a slash, e.g. `ICMP 3/3` for a port unreachable message.

26.5.9. Building the Rule Script with Symbolic Substitution

Some experienced IPF users create a file containing the rules and code them in a manner compatible with running them as a script with symbolic substitution. The major benefit of doing this is that you only have to change the value associated with the symbolic name and when the script is run all the rules containing the symbolic name will have the value substituted in the rules. Being a script, you can use symbolic substitution to code frequently used values and substitute them in multiple rules. You will see this in the following example.

The script syntax used here is compatible with the `sh`, `csh`, and `tcsh` shells.

Symbolic substitution fields are prefixed with a dollar sign: `$`.

Symbolic fields do not have the `$` prefix.

The value to populate the symbolic field must be enclosed with double quotes (`"`).

Start your rule file with something like this:

```
##### Start of IPF rules script #####

oif="dc0"           # name of the outbound interface
odns="192.0.2.11"   # ISP's DNS server IP address
myip="192.0.2.7"    # my static IP address from ISP
ks="keep state"
fks="flags S keep state"

# You can choose between building /etc/ipf.rules file
# from this script or running this script "as is".
#
# Uncomment only one line and comment out another.
#
# 1) This can be used for building /etc/ipf.rules:
#cat > /etc/ipf.rules << EOF
#
# 2) This can be used to run script "as is":
/sbin/ipf -Fa -f - << EOF

# Allow out access to my ISP's Domain name server.
pass out quick on $oif proto tcp from any to $odns port = 53 $fks
pass out quick on $oif proto udp from any to $odns port = 53 $fks

# Allow out non-secure standard www function
pass out quick on $oif proto tcp from $myip to any port = 80 $fks

# Allow out secure www function https over TLS SSL
pass out quick on $oif proto tcp from $myip to any port = 443 $fks
EOF
##### End of IPF rules script #####
```

That is all there is to it. The rules are not important in this example; how the symbolic substitution fields are populated and used are. If the above example was in a file named `/etc/ipf.rules.script`, you could reload these rules by entering the following command:

```
# sh /etc/ipf.rules.script
```

There is one problem with using a rules file with embedded symbolics: IPF does not understand symbolic substitution, and cannot read such scripts directly.

This script can be used in one of two ways:

- Uncomment the line that begins with `cat`, and comment out the line that begins with `/sbin/ipf`. Place `ipfilter_enable="YES"` into `/etc/rc.conf` as usual, and run script once after each modification to create or update `/etc/ipf.rules`.
- Disable IPFILTER in system startup scripts by adding `ipfilter_enable="NO"` (this is default value) into `/etc/rc.conf` file.

Add a script like the following to your `/usr/local/etc/rc.d/` startup directory. The script should have an obvious name like `ipf.loadrules.sh`. The `.sh` extension is mandatory.

```
#!/bin/sh
sh /etc/ipf.rules.script
```

The permissions on this script file must be read, write, execute for owner `root`.

```
# chmod 700 /usr/local/etc/rc.d/ipf.loadrules.sh
```

Now, when your system boots, your IPF rules will be loaded.

26.5.10. IPF Rule Sets

A rule set is a group of ipf rules coded to pass or block packets based on the values contained in the packet. The bi-directional exchange of packets between hosts comprises a session conversation. The firewall rule set processes the packet two times, once on its arrival from the public Internet host and again as it leaves for its return trip back to the public Internet host. Each TCP/IP service (i.e. telnet, www, mail, etc.) is predefined by its protocol, source and destination IP address, or the source and destination port number. This is the basic selection criteria used to create rules which will pass or block services.

IPF was originally written using a rules processing logic of „the last matching rule wins” and used only stateless rules. Over time IPF has been enhanced to include a „quick” option and a stateful „keep state” option which drastically modernized the rule processing logic.

The instructions contained in this section are based on using rules that contain the „quick” option and the stateful „keep state” option. This is the basic framework for coding an inclusive firewall rule set.

An inclusive firewall only allows services matching the rules through. This way you can control what services can originate behind the firewall destined for the public Internet and also control the services which can originate from the public Internet accessing your private network. Everything else is blocked and logged by default design. Inclusive firewalls are much, much securer than exclusive firewall rule sets and is the only rule set type covered herein.

Ostrzeżenie When working with the firewall rules, be *very careful*. Some configurations *will lock you out* of the server. To be on the safe side, you may wish to consider performing the initial firewall configuration from the local console rather than doing it remotely e.g. via **ssh**.

26.5.11. Rule Syntax

The rule syntax presented here has been simplified to only address the modern stateful rule context and „first matching rule wins” logic. For the complete legacy rule syntax description see the ipf(8) manual page.

A # character is used to mark the start of a comment and may appear at the end of a rule line or on its own line. Blank lines are ignored.

Rules contain keywords. These keywords have to be coded in a specific order from left to right on the line. Keywords are identified in bold type. Some keywords have sub-options which may be keywords themselves and also include more sub-options. Each of the headings in the below syntax has a bold section header which expands on the content.

```
ACTION IN-OUT OPTIONS SELECTION STATEFUL PROTO SRC_ADDR,DST_ADDR OBJECT PORT_NUM
TCP_FLAG STATEFUL
```

```
ACTION = block | pass
```

```
IN-OUT = in | out
```

OPTIONS = log | quick | on interface-name

SELECTION = proto value | source/destination IP | port = number | flags flag-value

PROTO = tcp/udp | udp | tcp | icmp

SRC_ADD, DST_ADDR = all | from object to object

OBJECT = IP address | any

PORT_NUM = port number

TCP_FLAG = S

STATEFUL = keep state

26.5.11.1. ACTION

The action indicates what to do with the packet if it matches the rest of the filter rule. Each rule *must* have a action. The following actions are recognized:

`block` indicates that the packet should be dropped if the selection parameters match the packet.

`pass` indicates that the packet should exit the firewall if the selection parameters match the packet.

26.5.11.2. IN-OUT

A mandatory requirement is that each filter rule explicitly state which side of the I/O it is to be used on. The next keyword must be either `in` or `out` and one or the other has to be coded or the rule will not pass syntax checks.

`in` means this rule is being applied against an inbound packet which has just been received on the interface facing the public Internet.

`out` means this rule is being applied against an outbound packet destined for the interface facing the public Internet.

26.5.11.3. OPTIONS

Notatka: These options must be used in the order shown here.

`log` indicates that the packet header will be written to the `ipl log` (as described in the LOGGING section below) if the selection parameters match the packet.

`quick` indicates that if the selection parameters match the packet, this rule will be the last rule checked, allowing a „short-circuit” path to avoid processing any following rules for this packet. This option is a mandatory requirement for the modernized rules processing logic.

`on` indicates the interface name to be incorporated into the selection parameters. Interface names are as displayed by `ifconfig(8)`. Using this option, the rule will only match if the packet is going through that interface in the specified direction (in/out). This option is a mandatory requirement for the modernized rules processing logic.

When a packet is logged, the headers of the packet are written to the IPL packet logging pseudo-device. Immediately following the `log` keyword, the following qualifiers may be used (in this order):

`body` indicates that the first 128 bytes of the packet contents will be logged after the headers.

first If the `log` keyword is being used in conjunction with a „keep state” option, it is recommended that this option is also applied so that only the triggering packet is logged and not every packet which thereafter matches the „keep state” information.

26.5.11.4. SELECTION

The keywords described in this section are used to describe attributes of the packet to be interrogated when determining whether rules match or not. There is a keyword `subject`, and it has sub-option keywords, one of which has to be selected. The following general-purpose attributes are provided for matching, and must be used in this order:

26.5.11.5. PROTO

`proto` is the subject keyword and must be coded along with one of its corresponding keyword sub-option values. The value allows a specific protocol to be matched against. This option is a mandatory requirement for the modernized rules processing logic.

`tcp/udp` | `udp` | `tcp` | `icmp` or any protocol names found in `/etc/protocols` are recognized and may be used. The special protocol keyword `tcp/udp` may be used to match either a TCP or a UDP packet, and has been added as a convenience to save duplication of otherwise identical rules.

26.5.11.6. SRC_ADDR/DST_ADDR

The `all` keyword is essentially a synonym for „from any to any” with no other match parameters.

from `src` to `dst`: the `from` and `to` keywords are used to match against IP addresses. Rules must specify BOTH source and destination parameters. `any` is a special keyword that matches any IP address. Examples of use: „from any to any” or „from 0.0.0.0/0 to any” or „from any to 0.0.0.0/0” or „from 0.0.0.0 to any” or „from any to 0.0.0.0”.

IP addresses may be specified as a dotted IP address numeric form/mask-length, or as single dotted IP address numeric form.

There is no way to match ranges of IP addresses which do not express themselves easily as mask-length. See this web page for help on writing mask-length: <http://jodies.de/ipcalc>.

26.5.11.7. PORT

If a port match is included, for either or both of source and destination, then it is only applied to TCP and UDP packets. When composing port comparisons, either the service name from `/etc/services` or an integer port number may be used. When the port appears as part of the `from` object, it matches the source port number; when it appears as part of the `to` object, it matches the destination port number. The use of the port option with the `to` object is a mandatory requirement for the modernized rules processing logic. Example of use: „from any to any port = 80”

Port comparisons may be done in a number of forms, with a number of comparison operators, or port ranges may be specified.

port `"="` | `"!="` | `"<"` | `">"` | `"<="` | `">="` | `"eq"` | `"ne"` | `"lt"` | `"gt"` | `"le"` | `"ge"`.

To specify port ranges, port `"<>"` | `"><"`

Ostrzeżenie Following the source and destination matching parameters, the following two parameters are mandatory requirements for the modernized rules processing logic.

26.5.11.8. TCP_FLAG

Flags are only effective for TCP filtering. The letters represents one of the possible flags that can be interrogated in the TCP packet header.

The modernized rules processing logic uses the `flags s` parameter to identify the tcp session start request.

26.5.11.9. STATEFUL

`keep state` indicates that on a pass rule, any packets that match the rules selection parameters should activate the stateful filtering facility.

Notatka: This option is a mandatory requirement for the modernized rules processing logic.

26.5.12. Stateful Filtering

Stateful filtering treats traffic as a bi-directional exchange of packets comprising a session conversation. When activated, keep-state dynamically generates internal rules for each anticipated packet being exchanged during the bi-directional session conversation. It has the interrogation abilities to determine if the session conversation between the originating sender and the destination are following the valid procedure of bi-directional packet exchange. Any packets that do not properly fit the session conversation template are automatically rejected as impostors.

Keep state will also allow ICMP packets related to a TCP or UDP session through. So if you get ICMP type 3 code 4 in response to some web surfing allowed out by a keep state rule, they will be automatically allowed in. Any packet that IPF can be certain is part of an active session, even if it is a different protocol, will be let in.

What happens is:

Packets destined to go out the interface connected to the public Internet are first checked against the dynamic state table, if the packet matches the next expected packet comprising in a active session conversation, then it exits the firewall and the state of the session conversation flow is updated in the dynamic state table, the remaining packets get checked against the outbound rule set.

Packets coming in to the interface connected to the public Internet are first checked against the dynamic state table, if the packet matches the next expected packet comprising a active session conversation, then it exits the firewall and the state of the session conversation flow is updated in the dynamic state table, the remaining packets get checked against the inbound rule set.

When the conversation completes it is removed from the dynamic state table.

Stateful filtering allows you to focus on blocking/passing new sessions. If the new session is passed, all its subsequent packets will be allowed through automatically and any impostors automatically rejected. If a new session is blocked, none of its subsequent packets will be allowed through. Stateful filtering has technically advanced interrogation abilities capable of defending against the flood of different attack methods currently employed by attackers.

26.5.13. Inclusive Rule Set Example

The following rule set is an example of how to code a very secure inclusive type of firewall. An inclusive firewall only allows services matching pass rules through and blocks all other by default. All firewalls have at the minimum two interfaces which have to have rules to allow the firewall to function.

All UNIX flavored systems including FreeBSD are designed to use interface `lo0` and IP address `127.0.0.1` for internal communication within the operating system. The firewall rules must contain rules to allow free unmolested movement of these special internally used packets.

The interface which faces the public Internet is the one where you place your rules to authorize and control access out to the public Internet and access requests arriving from the public Internet. This can be your user PPP `tun0` interface or your NIC that is connected to your DSL or cable modem.

In cases where one or more NICs are cabled to private LANs behind the firewall, those interfaces must have a rule coded to allow free unmolested movement of packets originating from those LAN interfaces.

The rules should be first organized into three major sections: all the free unmolested interfaces, the public interface outbound, and the public interface inbound.

The rules in each of the public interface sections should have the most frequently matched rules placed before less commonly matched rules, with the last rule in the section blocking and logging all packets on that interface and direction.

The Outbound section in the following rule set only contains 'pass' rules which contain selection values that uniquely identify the service that is authorized for public Internet access. All the rules have the 'quick', 'on', 'proto', 'port', and 'keep state' option coded. The 'proto tcp' rules have the 'flag' option included to identify the session start request as the triggering packet to activate the stateful facility.

The Inbound section has all the blocking of undesirable packets first, for two different reasons. The first is that these things being blocked may be part of an otherwise valid packet which may be allowed in by the later authorized service rules. The second reason is that by having a rule that explicitly blocks selected packets that I receive on an infrequent basis and that I do not want to see in the log, they will not be caught by the last rule in the section which blocks and logs all packets which have fallen through the rules. The last rule in the section which blocks and logs all packets is how you create the legal evidence needed to prosecute the people who are attacking your system.

Another thing you should take note of, is there is no response returned for any of the undesirable stuff, their packets just get dropped and vanish. This way the attacker has no knowledge if his packets have reached your system. The less the attackers can learn about your system, the more time they must invest before actually doing something bad. The inbound 'nmap OS fingerprint' attempts rule I log the first occurrence because this is something an attacker would do.

Any time you see log messages on a rule with 'log first'. You should do an `ipfstat -hio` command to see the number of times the rule has been matched so you know if you are being flooded, i.e. under attack.

When you log packets with port numbers you do not recognize, look it up in `/etc/services` or go to <http://www.securitystats.com/tools/portsearch.php> and do a port number lookup to find what the purpose of that port number is.

Check out this link for port numbers used by Trojans <http://www.simovits.com/trojans/trojans.html>.

The following rule set is a complete very secure 'inclusive' type of firewall rule set that I have used on my system. You can not go wrong using this rule set for your own. Just comment out any pass rules for services that you do not want to authorize.

If you see messages in your log that you want to stop seeing just add a block rule in the inbound section.

You have to change the `dc0` interface name in every rule to the interface name of the Nic card that connects your system to the public Internet. For user PPP it would be `tun0`.

Add the following statements to `/etc/ipf.rules`:

```
#####
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN
#####

#pass out quick on xl0 all
#pass in quick on xl0 all

#####
# No restrictions on Loopback Interface
#####
pass in quick on lo0 all
pass out quick on lo0 all

#####
# Interface facing Public Internet (Outbound Section)
# Interrogate session start requests originating from behind the
# firewall on the private network
# or from this gateway server destined for the public Internet.
#####

# Allow out access to my ISP's Domain name server.
# xxx must be the IP address of your ISP's DNS.
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
pass out quick on dc0 proto tcp from any to xxx port = 53 flags S keep state
pass out quick on dc0 proto udp from any to xxx port = 53 keep state

# Allow out access to my ISP's DHCP server for cable or DSL networks.
# This rule is not needed for 'user ppp' type connection to the
# public Internet, so you can delete this whole group.
# Use the following rule and check log for IP address.
# Then put IP address in commented out rule & delete first rule
pass out log quick on dc0 proto udp from any to any port = 67 keep state
#pass out quick on dc0 proto udp from any to z.z.z.z port = 67 keep state

# Allow out non-secure standard www function
pass out quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Allow out secure www function https over TLS SSL
pass out quick on dc0 proto tcp from any to any port = 443 flags S keep state

# Allow out send & get email function
pass out quick on dc0 proto tcp from any to any port = 110 flags S keep state
pass out quick on dc0 proto tcp from any to any port = 25 flags S keep state

# Allow out Time
```

```

pass out quick on dc0 proto tcp from any to any port = 37 flags S keep state

# Allow out nntp news
pass out quick on dc0 proto tcp from any to any port = 119 flags S keep state

# Allow out gateway & LAN users non-secure FTP ( both passive & active modes)
# This function uses the IPNAT built in FTP proxy function coded in
# the nat rules file to make this single rule function correctly.
# If you want to use the pkg_add command to install application packages
# on your gateway system you need this rule.
pass out quick on dc0 proto tcp from any to any port = 21 flags S keep state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
pass out quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Allow out non-secure Telnet
pass out quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Allow out FBSD CVSUP function
pass out quick on dc0 proto tcp from any to any port = 5999 flags S keep state

# Allow out ping to public Internet
pass out quick on dc0 proto icmp from any to any icmp-type 8 keep state

# Allow out whois for LAN PC to public Internet
pass out quick on dc0 proto tcp from any to any port = 43 flags S keep state

# Block and log only the first occurrence of everything
# else that's trying to get out.
# This rule enforces the block all by default logic.
block out log first quick on dc0 all

#####
# Interface facing Public Internet (Inbound Section)
# Interrogate packets originating from the public Internet
# destined for this gateway server or the private network.
#####

# Block all inbound traffic from non-routable or reserved address spaces
block in quick on dc0 from 192.168.0.0/16 to any      #RFC 1918 private IP
block in quick on dc0 from 172.16.0.0/12 to any      #RFC 1918 private IP
block in quick on dc0 from 10.0.0.0/8 to any         #RFC 1918 private IP
block in quick on dc0 from 127.0.0.0/8 to any        #loopback
block in quick on dc0 from 0.0.0.0/8 to any          #loopback
block in quick on dc0 from 169.254.0.0/16 to any     #DHCP auto-config
block in quick on dc0 from 192.0.2.0/24 to any       #reserved for docs
block in quick on dc0 from 204.152.64.0/23 to any    #Sun cluster interconnect
block in quick on dc0 from 224.0.0.0/3 to any        #Class D & E multicast

##### Block a bunch of different nasty things. #####
# That I do not want to see in the log

```

```

# Block frags
block in quick on dc0 all with frags

# Block short tcp packets
block in quick on dc0 proto tcp all with short

# block source routed packets
block in quick on dc0 all with opt lsrr
block in quick on dc0 all with opt ssrr

# Block nmap OS fingerprint attempts
# Log first occurrence of these so I can get their IP address
block in log first quick on dc0 proto tcp from any to any flags FUP

# Block anything with special options
block in quick on dc0 all with ipopts

# Block public pings
block in quick on dc0 proto icmp all icmp-type 8

# Block ident
block in quick on dc0 proto tcp from any to any port = 113

# Block all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
block in log first quick on dc0 proto tcp/udp from any to any port = 137
block in log first quick on dc0 proto tcp/udp from any to any port = 138
block in log first quick on dc0 proto tcp/udp from any to any port = 139
block in log first quick on dc0 proto tcp/udp from any to any port = 81

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it's the only
# authorized source to send this packet type. Only necessary for
# cable or DSL configurations. This rule is not needed for
# 'user ppp' type connection to the public Internet.
# This is the same IP address you captured and
# used in the outbound section.
pass in quick on dc0 proto udp from z.z.z.z to any port = 68 keep state

# Allow in standard www function because I have apache server
pass in quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID/PW passed over public Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
#pass in quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Allow in secure FTP, Telnet, and SCP from public Internet
# This function is using SSH (secure shell)
pass in quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Block and log only first occurrence of all remaining traffic

```

```
# coming into the firewall. The logging of only the first
# occurrence stops a .denial of service. attack targeted
# at filling up your log file space.
# This rule enforces the block all by default logic.
block in log first quick on dc0 all
##### End of rules file #####
```

26.5.14. NAT

NAT stands for Network Address Translation. To those familiar with Linux, this concept is called IP Masquerading; NAT and IP Masquerading are the same thing. One of the many things the IPF NAT function enables is the ability to have a private Local Area Network (LAN) behind the firewall sharing a single ISP assigned IP address on the public Internet.

You may ask why would someone want to do this. ISPs normally assign a dynamic IP address to their non-commercial users. Dynamic means that the IP address can be different each time you dial in and log on to your ISP, or for cable and DSL modem users when you power off and then power on your modems you can get assigned a different IP address. This IP address is how you are known to the public Internet.

Now let's say you have five PCs at home and each one needs Internet access. You would have to pay your ISP for an individual Internet account for each PC and have five phone lines.

With NAT you only need a single account with your ISP, then cable your other four PCs to a switch and the switch to the NIC in your FreeBSD system which is going to service your LAN as a gateway. NAT will automatically translate the private LAN IP address for each separate PC on the LAN to the single public IP address as it exits the firewall bound for the public Internet. It also does the reverse translation for returning packets.

NAT is most often accomplished without the approval, or knowledge, of your ISP and in most cases is grounds for your ISP terminating your account if found out. Commercial users pay a lot more for their Internet connection and usually get assigned a block of static IP address which never change. The ISP also expects and consents to their Commercial customers using NAT for their internal private LANs.

There is a special range of IP addresses reserved for NATed private LAN IP address. According to RFC 1918, you can use the following IP ranges for private nets which will never be routed directly to the public Internet:

Start IP 10.0.0.0	-	Ending IP 10.255.255.255
Start IP 172.16.0.0	-	Ending IP 172.31.255.255
Start IP 192.168.0.0	-	Ending IP 192.168.255.255

26.5.15. IPNAT

NAT rules are loaded by using the `ipnat` command. Typically the NAT rules are stored in `/etc/ipnat.rules`. See `ipnat(1)` for details.

When changing the NAT rules after NAT has been started, make your changes to the file containing the NAT rules, then run `ipnat` command with the `-CF` flags to delete the internal in use NAT rules and flush the contents of the translation table of all active entries.

To reload the NAT rules issue a command like this:

```
# ipnat -CF -f /etc/ipnat.rules
```

To display some statistics about your NAT, use this command:

```
# ipnat -s
```

To list the NAT table's current mappings, use this command:

```
# ipnat -l
```

To turn verbose mode on, and display information relating to rule processing and active rules/table entries:

```
# ipnat -v
```

26.5.16. IPNAT Rules

NAT rules are very flexible and can accomplish many different things to fit the needs of commercial and home users.

The rule syntax presented here has been simplified to what is most commonly used in a non-commercial environment. For a complete rule syntax description see the `ipnat(5)` manual page.

The syntax for a NAT rule looks something like this:

```
map IF LAN_IP_RANGE -> PUBLIC_ADDRESS
```

The keyword `map` starts the rule.

Replace `IF` with the external interface.

The `LAN_IP_RANGE` is what your internal clients use for IP Addressing, usually this is something like `192.168.1.0/24`.

The `PUBLIC_ADDRESS` can either be the external IP address or the special keyword `0/32`, which means to use the IP address assigned to `IF`.

26.5.17. How NAT works

A packet arrives at the firewall from the LAN with a public destination. It passes through the outbound filter rules, NAT gets his turn at the packet and applies its rules top down, first matching rule wins. NAT tests each of its rules against the packets interface name and source IP address. When a packets interface name matches a NAT rule then the [source IP address, i.e. private LAN IP address] of the packet is checked to see if it falls within the IP address range specified to the left of the arrow symbol on the NAT rule. On a match the packet has its source IP address rewritten with the public IP address obtained by the `0/32` keyword. NAT posts a entry in its internal NAT table so when the packet returns from the public Internet it can be mapped back to its original private IP address and then passed to the filter rules for processing.

26.5.18. Enabling IPNAT

To enable IPNAT add these statements to `/etc/rc.conf`.

To enable your machine to route traffic between interfaces:

```
gateway_enable="YES"
```

To start IPNAT automatically each time:

```
ipnat_enable="YES"
```

To specify where to load the IPNAT rules from:

```
ipnat_rules="/etc/ipnat.rules"
```

26.5.19. NAT for a very large LAN

For networks that have large numbers of PC's on the LAN or networks with more than a single LAN, the process of funneling all those private IP addresses into a single public IP address becomes a resource problem that may cause problems with the same port numbers being used many times across many NATed LAN PC's, causing collisions. There are two ways to relieve this resource problem.

26.5.19.1. Assigning Ports to Use

A normal NAT rule would look like:

```
map dc0 192.168.1.0/24 -> 0/32
```

In the above rule the packet's source port is unchanged as the packet passes through IPNAT. By adding the portmap keyword you can tell IPNAT to only use source ports in a range. For example the following rule will tell IPNAT to modify the source port to be within that range:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

Additionally we can make things even easier by using the auto keyword to tell IPNAT to determine by itself which ports are available to use:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

26.5.19.2. Using a pool of public addresses

In very large LANs there comes a point where there are just too many LAN addresses to fit into a single public address. If a block of public IP addresses is available, you can use these addresses as a „pool”, and let IPNAT pick one of the public IP addresses as packet-addresses are mapped on their way out.

For example, instead of mapping all packets through a single public IP address, as in:

```
map dc0 192.168.1.0/24 -> 204.134.75.1
```

A range of public IP addresses can be specified either with a netmask:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
```

or using CIDR notation:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/24
```

26.5.20. Port Redirection

A very common practice is to have a web server, email server, database server and DNS server each segregated to a different PC on the LAN. In this case the traffic from these servers still have to be NATed, but there has to be some way to direct the inbound traffic to the correct LAN PCs. IPNAT has the redirection facilities of NAT to solve this problem. Lets say you have your web server on LAN address 10.0.10.25 and your single public IP address is 20.20.20.5 you would code the rule like this:

```
rdr dc0 20.20.20.5/32 port 80 -> 10.0.10.25 port 80
```

or:

```
rdr dc0 0/32 port 80 -> 10.0.10.25 port 80
```

or for a LAN DNS Server on LAN address of 10.0.10.33 that needs to receive public DNS requests:

```
rdr dc0 20.20.20.5/32 port 53 -> 10.0.10.33 port 53 udp
```

26.5.21. FTP and NAT

FTP is a dinosaur left over from the time before the Internet as it is known today, when research universities were leased lined together and FTP was used to share files among research Scientists. This was a time when data security was not a consideration. Over the years the FTP protocol became buried into the backbone of the emerging Internet and its username and password being sent in clear text was never changed to address new security concerns. FTP has two flavors, it can run in active mode or passive mode. The difference is in how the data channel is acquired. Passive mode is more secure as the data channel is acquired by the ordinal ftp session requester. For a real good explanation of FTP and the different modes see <http://www.slacksite.com/other/ftp.html>.

26.5.21.1. IPNAT Rules

IPNAT has a special built in FTP proxy option which can be specified on the NAT map rule. It can monitor all outbound packet traffic for FTP active or passive start session requests and dynamically create temporary filter rules containing only the port number really in use for the data channel. This eliminates the security risk FTP normally exposes the firewall to from having large ranges of high order port numbers open.

This rule will handle all the traffic for the internal LAN:

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
```

This rule handles the FTP traffic from the gateway:

```
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
```

This rule handles all non-FTP traffic from the internal LAN:

```
map dc0 10.0.10.0/29 -> 0/32
```

The FTP map rule goes before our regular map rule. All packets are tested against the first rule from the top. Matches on interface name, then private LAN source IP address, and then is it a FTP packet. If all that matches then the special FTP proxy creates temp filter rules to let the FTP session packets pass in and out, in addition to also NATing

the FTP packets. All LAN packets that are not FTP do not match the first rule and fall through to the third rule and are tested, matching on interface and source IP, then are NATed.

26.5.21.2. IPNAT FTP Filter Rules

Only one filter rule is needed for FTP if the NAT FTP proxy is used.

Without the FTP Proxy you will need the following three rules:

```
# Allow out LAN PC client FTP to public Internet
# Active and passive modes
pass out quick on rl0 proto tcp from any to any port = 21 flags S keep state

# Allow out passive mode data channel high order port numbers
pass out quick on rl0 proto tcp from any to any port > 1024 flags S keep state

# Active mode let data channel in from FTP server
pass in quick on rl0 proto tcp from any to any port = 20 flags S keep state
```

26.5.21.3. FTP NAT Proxy Bug

As of IPFILTER version 3.4.31 the FTP proxy works as documented during the FTP session until the session is told to close. When the close happens packets returning from the remote FTP server are blocked and logged coming in on port 21. The NAT FTP/proxy appears to remove its temp rules prematurely, before receiving the response from the remote FTP server acknowledging the close. A problem report was posted to the IPF mailing list.

The solution is to add a filter rule to get rid of these unwanted log messages or do nothing and ignore FTP inbound error messages in your log. Most people do not use outbound FTP too often.

```
block in quick on rl0 proto tcp from any to any port = 21
```

26.6. IPFW

Notatka: This section is work in progress. The contents might not be accurate at all times.

The IPFIREWALL (IPFW) is a FreeBSD sponsored firewall software application authored and maintained by FreeBSD volunteer staff members. It uses the legacy stateless rules and a legacy rule coding technique to achieve what is referred to as Simple Stateful logic.

The IPFW sample rule set (found in `/etc/rc.firewall`) in the standard FreeBSD install is rather simple and it is not expected that it used directly without modifications. The example does not use stateful filtering, which is beneficial in most setups, so it will not be used as base for this section.

The IPFW stateless rule syntax is empowered with technically sophisticated selection capabilities which far surpasses the knowledge level of the customary firewall installer. IPFW is targeted at the professional user or the advanced technical computer hobbyist who have advanced packet selection requirements. A high degree of detailed

knowledge into how different protocols use and create their unique packet header information is necessary before the power of the IPFW rules can be unleashed. Providing that level of explanation is out of the scope of this section of the handbook.

IPFW is composed of seven components, the primary component is the kernel firewall filter rule processor and its integrated packet accounting facility, the logging facility, the 'divert' rule which triggers the NAT facility, and the advanced special purpose facilities, the dummynet traffic shaper facilities, the 'fwd rule' forward facility, the bridge facility, and the ipstealth facility.

26.6.1. Enabling IPFW

IPFW is included in the basic FreeBSD install as a separate run time loadable module. The system will dynamically load the kernel module when the `rc.conf` statement `firewall_enable="YES"` is used. You do not need to compile IPFW into the FreeBSD kernel unless you want NAT function enabled.

After rebooting your system with `firewall_enable="YES"` in `rc.conf` the following white highlighted message is displayed on the screen as part of the boot process:

```
ipfw2 initialized, divert disabled, rule-based forwarding disabled, default to deny, logging disabled
```

The loadable module does have logging ability compiled in. To enable logging and set the verbose logging limit, there is a knob you can set in `/etc/sysctl.conf` by adding these statements, logging will be enabled on future reboots:

```
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=5
```

26.6.2. Kernel Options

It is not a mandatory requirement that you enable IPFW by compiling the following options into the FreeBSD kernel unless you need NAT function. It is presented here as background information.

```
options      IPFIREWALL
```

This option enables IPFW as part of the kernel

```
options      IPFIREWALL_VERBOSE
```

Enables logging of packets that pass through IPFW and have the 'log' keyword specified in the rule set.

```
options      IPFIREWALL_VERBOSE_LIMIT=5
```

Limits the number of packets logged through syslogd(8) on a per entry basis. You may wish to use this option in hostile environments which you want to log firewall activity. This will close a possible denial of service attack via syslog flooding.

```
options      IPFIREWALL_DEFAULT_TO_ACCEPT
```

This option will allow everything to pass through the firewall by default, which is a good idea when you are first setting up your firewall.

```
options    IPV6FIREWALL
options    IPV6FIREWALL_VERBOSE
options    IPV6FIREWALL_VERBOSE_LIMIT
options    IPV6FIREWALL_DEFAULT_TO_ACCEPT
```

These options are exactly the same as the IPv4 options but they are for IPv6. If you do not use IPv6 you might want to use `IPV6FIREWALL` without any rules to block all IPv6

```
options    IPDIVERT
```

This enables the use of NAT functionality.

Notatka: If you do not include `IPFIREWALL_DEFAULT_TO_ACCEPT` or set your rules to allow incoming packets you will block all packets going to and from this machine.

26.6.3. `/etc/rc.conf` Options

If you do not have IPFW compiled into your kernel you will need to load it with the following statement in your `/etc/rc.conf`:

```
firewall_enable="YES"
```

To select one of the default firewall types provided by FreeBSD, select one by reading the `/etc/rc.firewall` file and place it in the following:

```
firewall_type="open"
```

Or load custom rules by setting the following variable to the file containing them:

```
firewall_script="/etc/ipfw.rules"
```

Enable logging:

```
firewall_logging="YES"
```

Ostrzeżenie The only thing that the `firewall_logging` variable will do is setting the `net.inet.ip.fw.verbose` `sysctl` variable to the value of 1 (see [Sekcja 26.6.1](#)). There is no `rc.conf` variable to set log limitations, but it can be set via `sysctl` variable, manually or from the `/etc/sysctl.conf` file:

```
net.inet.ip.fw.verbose_limit=5
```

If your machine is acting as a gateway, i.e. providing Network Address Translation (NAT) via `natd(8)`, please refer to [Sekcja 27.8](#) for information regarding the required `/etc/rc.conf` options.

26.6.4. The IPFW Command

The `ipfw` command is the normal vehicle for making manual single rule additions or deletions to the firewall active internal rules while it is running. The problem with using this method is once your system is shutdown or halted all the rules you added or changed or deleted are lost. Writing all your rules in a file and using that file to load the rules at boot time, or to replace in mass the currently running firewall rules with changes you made to the files content is the recommended method used here.

The `ipfw` command is still a very useful to display the running firewall rules to the console screen. The IPFW accounting facility dynamically creates a counter for each rule that counts each packet that matches the rule. During the process of testing a rule, listing the rule with its counter is the one of the ways of determining if the rule is functioning.

To list all the rules in sequence:

```
# ipfw list
```

To list all the rules with a time stamp of when the last time the rule was matched:

```
# ipfw -t list
```

To list the accounting information, packet count for matched rules along with the rules themselves. The first column is the rule number, followed by the number of outgoing matched packets, followed by the number of incoming matched packets, and then the rule itself.

```
# ipfw -a list
```

List the dynamic rules in addition to the static rules:

```
# ipfw -d list
```

Also show the expired dynamic rules:

```
# ipfw -d -e list
```

Zero the counters:

```
# ipfw zero
```

Zero the counters for just rule *NUM*:

```
# ipfw zero NUM
```

26.6.5. IPFW Rule Sets

A rule set is a group of `ipfw` rules coded to allow or deny packets based on the values contained in the packet. The bi-directional exchange of packets between hosts comprises a session conversation. The firewall rule set processes the packet twice: once on its arrival from the public Internet host and again as it leaves for its return trip back to the public Internet host. Each tcp/ip service (i.e. telnet, www, mail, etc.) is predefined by its protocol, and port number. This is the basic selection criteria used to create rules which will allow or deny services.

When a packet enters the firewall it is compared against the first rule in the rule set and progress one rule at a time moving from top to bottom of the set in ascending rule number sequence order. When the packet matches a rule

selection parameters, the rules action field value is executed and the search of the rule set terminates for that packet. This is referred to as „the first match wins” search method. If the packet does not match any of the rules, it gets caught by the mandatory ipfw default rule, number 65535 which denies all packets and discards them without any reply back to the originating destination.

Notatka: The search continues after `count`, `skipto` and `tee` rules.

The instructions contained here are based on using rules that contain the stateful 'keep state', 'limit', 'in'/'out', and via options. This is the basic framework for coding an inclusive type firewall rule set.

An inclusive firewall only allows services matching the rules through. This way you can control what services can originate behind the firewall destined for the public Internet and also control the services which can originate from the public Internet accessing your private network. Everything else is denied by default design. Inclusive firewalls are much, much more secure than exclusive firewall rule sets and is the only rule set type covered here in.

Ostrzeżenie When working with the firewall rules be careful, you can end up locking your self out.

26.6.5.1. Rule Syntax

The rule syntax presented here has been simplified to what is necessary to create a standard inclusive type firewall rule set. For a complete rule syntax description see the ipfw(8) manual page.

Rules contain keywords: these keywords have to be coded in a specific order from left to right on the line. Keywords are identified in bold type. Some keywords have sub-options which may be keywords themselves and also include more sub-options.

is used to mark the start of a comment and may appear at the end of a rule line or on its own lines. Blank lines are ignored.

```
CMD RULE_NUMBER ACTION LOGGING SELECTION STATEFUL
```

26.6.5.1.1. CMD

Each new rule has to be prefixed with `add` to add the rule to the internal table.

26.6.5.1.2. RULE_NUMBER

Each rule has to have a rule number to go with it.

26.6.5.1.3. ACTION

A rule can be associated with one of the following actions, which will be executed when the packet matches the selection criterion of the rule.

```
allow | accept | pass | permit
```

These all mean the same thing which is to allow packets that match the rule to exit the firewall rule processing. The search terminates at this rule.

check-state

Checks the packet against the dynamic rules table. If a match is found, execute the action associated with the rule which generated this dynamic rule, otherwise move to the next rule. The check-state rule does not have selection criterion. If no check-state rule is present in the rule set, the dynamic rules table is checked at the first keep-state or limit rule.

deny | drop

Both words mean the same thing which is to discard packets that match this rule. The search terminates.

26.6.5.1.4. Logging*log or logamount*

When a packet matches a rule with the log keyword, a message will be logged to syslogd with a facility name of SECURITY. The logging only occurs if the number of packets logged so far for that particular rule does not exceed the logamount parameter. If no logamount is specified, the limit is taken from the sysctl variable net.inet.ip.fw.verbose_limit. In both cases, a value of zero removes the logging limit. Once the limit is reached, logging can be re-enabled by clearing the logging counter or the packet counter for that rule, see the ipfw reset log command.

Notatka: Logging is done after all other packet matching conditions have been successfully verified, and before performing the final action (accept, deny) on the packet. It is up to you to decide which rules you want to enable logging on.

26.6.5.1.5. Selection

The keywords described in this section are used to describe attributes of the packet to be interrogated when determining whether rules match the packet or not. The following general-purpose attributes are provided for matching, and must be used in this order:

udp | tcp | icmp

or any protocol names found in `/etc/protocols` are recognized and may be used. The value specified is protocol to be matched against. This is a mandatory requirement.

from src to dst

The from and to keywords are used to match against IP addresses. Rules must specify BOTH source and destination parameters. any is a special keyword that matches any IP address. me is a special keyword that matches any IP address configured on an interface in your FreeBSD system to represent the PC the firewall is running on (i.e. this box) as in 'from me to any' or 'from any to me' or 'from 0.0.0.0/0 to any' or 'from any to 0.0.0.0/0' or 'from 0.0.0.0 to any' or 'from any to 0.0.0.0' or 'from me to 0.0.0.0'. IP addresses are specified as a dotted IP address numeric form/mask-length, or as single dotted IP address numeric form. This is a mandatory requirement. See this link for help on writing mask-lengths. <http://jodies.de/ipcalc>

port number

For protocols which support port numbers (such as TCP and UDP). It is mandatory that you code the port number of the service you want to match on. Service names (from `/etc/services`) may be used instead of numeric port values.

in | out

Matches incoming or outgoing packets, respectively. The *in* and *out* are keywords and it is mandatory that you code one or the other as part of your rule matching criterion.

via IF

Matches packets going through the interface specified by exact name. The *via* keyword causes the interface to always be checked as part of the match process.

setup

This is a mandatory keyword that identifies the session start request for TCP packets.

keep-state

This is a mandatory keyword. Upon a match, the firewall will create a dynamic rule, whose default behavior is to match bidirectional traffic between source and destination IP/port using the same protocol.

limit {src-addr | src-port | dst-addr | dst-port}

The firewall will only allow *N* connections with the same set of parameters as specified in the rule. One or more of source and destination addresses and ports can be specified. The 'limit' and 'keep-state' can not be used on same rule. Limit provides the same stateful function as 'keep-state' plus its own functions.

26.6.5.2. Stateful Rule Option

Stateful filtering treats traffic as a bi-directional exchange of packets comprising a session conversation. It has the interrogation abilities to determine if the session conversation between the originating sender and the destination are following the valid procedure of bi-directional packet exchange. Any packets that do not properly fit the session conversation template are automatically rejected as impostors.

'check-state' is used to identify where in the IPFW rules set the packet is to be tested against the dynamic rules facility. On a match the packet exits the firewall to continue on its way and a new rule is dynamic created for the next anticipated packet being exchanged during this bi-directional session conversation. On a no match the packet advances to the next rule in the rule set for testing.

The dynamic rules facility is vulnerable to resource depletion from a SYN-flood attack which would open a huge number of dynamic rules. To counter this attack, FreeBSD added another new option named *limit*. This option is used to limit the number of simultaneous session conversations by interrogating the rules source or destinations fields as directed by the *limit* option and using the packet's IP address found there, in a search of the open dynamic rules counting the number of times this rule and IP address combination occurred, if this count is greater than the value specified on the *limit* option, the packet is discarded.

26.6.5.3. Logging Firewall Messages

The benefits of logging are obvious: it provides the ability to review after the fact the rules you activated logging on which provides information like, what packets had been dropped, what addresses they came from, where they were going, giving you a significant edge in tracking down attackers.

Even with the logging facility enabled, IPFW will not generate any rule logging on its own. The firewall administrator decides what rules in the rule set he wants to log and adds the *log* verb to those rules. Normally only deny rules are logged, like the deny rule for incoming ICMP pings. It is very customary to duplicate the ipfw default

deny everything rule with the log verb included as your last rule in the rule set. This way you get to see all the packets that did not match any of the rules in the rule set.

Logging is a two edged sword, if you are not careful, you can lose yourself in the over abundance of log data and fill your disk up with growing log files. DoS attacks that fill up disk drives is one of the oldest attacks around. These log message are not only written to syslogd, but also are displayed on the root console screen and soon become very annoying.

The `IPFWALL_VERBOSE_LIMIT=5` kernel option limits the number of consecutive messages sent to the system logger syslogd, concerning the packet matching of a given rule. When this option is enabled in the kernel, the number of consecutive messages concerning a particular rule is capped at the number specified. There is nothing to be gained from 200 log messages saying the same identical thing. For instance, five consecutive messages concerning a particular rule would be logged to syslogd, the remainder identical consecutive messages would be counted and posted to the syslogd with a phrase like this:

```
last message repeated 45 times
```

All logged packets messages are written by default to `/var/log/security` file, which is defined in the `/etc/syslog.conf` file.

26.6.5.4. Building a Rule Script

Most experienced IPFW users create a file containing the rules and code them in a manner compatible with running them as a script. The major benefit of doing this is the firewall rules can be refreshed in mass without the need of rebooting the system to activate the new rules. This method is very convenient in testing new rules as the procedure can be executed as many times as needed. Being a script, you can use symbolic substitution to code frequent used values and substitution them in multiple rules. You will see this in the following example.

The script syntax used here is compatible with the 'sh', 'csh', 'tcsh' shells. Symbolic substitution fields are prefixed with a dollar sign \$. Symbolic fields do not have the \$ prefix. The value to populate the Symbolic field must be enclosed to "double quotes".

Start your rules file like this:

```
##### start of example ipfw rules script #####
#
ipfw -q -f flush      # Delete all rules
# Set defaults
oif="tun0"            # out interface
odns="192.0.2.11"     # ISP's DNS server IP address
cmd="ipfw -q add "    # build rule prefix
ks="keep-state"       # just too lazy to key this each time
$cmd 00500 check-state
$cmd 00502 deny all from any to any frag
$cmd 00501 deny tcp from any to any established
$cmd 00600 allow tcp from any to any 80 out via $oif setup $ks
$cmd 00610 allow tcp from any to $odns 53 out via $oif setup $ks
$cmd 00611 allow udp from any to $odns 53 out via $oif $ks
##### End of example ipfw rules script #####
```

That is all there is to it. The rules are not important in this example, how the Symbolic substitution field are populated and used are.

If the above example was in `/etc/ipfw.rules` file, you could reload these rules by entering on the command line.

```
# sh /etc/ipfw.rules
```

The `/etc/ipfw.rules` file could be located anywhere you want and the file could be named any thing you would like.

The same thing could also be accomplished by running these commands by hand:

```
# ipfw -q -f flush
# ipfw -q add check-state
# ipfw -q add deny all from any to any frag
# ipfw -q add deny tcp from any to any established
# ipfw -q add allow tcp from any to any 80 out via tun0 setup keep-state
# ipfw -q add allow tcp from any to 192.0.2.11 53 out via tun0 setup keep-state
# ipfw -q add 00611 allow udp from any to 192.0.2.11 53 out via tun0 keep-state
```

26.6.5.5. Stateful Ruleset

The following non-NATed rule set is an example of how to code a very secure 'inclusive' type of firewall. An inclusive firewall only allows services matching pass rules through and blocks all other by default. All firewalls have at the minimum two interfaces which have to have rules to allow the firewall to function.

All UNIX flavored operating systems, FreeBSD included, are designed to use interface `lo0` and IP address `127.0.0.1` for internal communication within the operating system. The firewall rules must contain rules to allow free unmolested movement of these special internally used packets.

The interface which faces the public Internet, is the one which you code your rules to authorize and control access out to the public Internet and access requests arriving from the public Internet. This can be your ppp `tun0` interface or your NIC that is connected to your DSL or cable modem.

In cases where one or more than one NIC are connected to a private LANs behind the firewall, those interfaces must have rules coded to allow free unmolested movement of packets originating from those LAN interfaces.

The rules should be first organized into three major sections, all the free unmolested interfaces, public interface outbound, and the public interface inbound.

The order of the rules in each of the public interface sections should be in order of the most used rules being placed before less often used rules with the last rule in the section being a block log all packets on that interface and direction.

The Outbound section in the following rule set only contains 'allow' rules which contain selection values that uniquely identify the service that is authorized for public Internet access. All the rules have the, proto, port, in/out, via and keep state option coded. The 'proto tcp' rules have the 'setup' option included to identify the start session request as the trigger packet to be posted to the keep state stateful table.

The Inbound section has all the blocking of undesirable packets first for two different reasons. First is these things being blocked may be part of an otherwise valid packet which may be allowed in by the later authorized service rules. Second reason is that by having a rule that explicitly blocks selected packets that I receive on an infrequent bases and do not want to see in the log, this keeps them from being caught by the last rule in the section which blocks and logs all packets which have fallen through the rules. The last rule in the section which blocks and logs all packets is how you create the legal evidence needed to prosecute the people who are attacking your system.

Another thing you should take note of, is there is no response returned for any of the undesirable stuff, their packets just get dropped and vanish. This way the attackers has no knowledge if his packets have reached your system. The less the attackers can learn about your system the more secure it is. When you log packets with port numbers you do not recognize, look the numbers up in `/etc/services/` or go to <http://www.securitystats.com/tools/portsearch.php> and do a port number lookup to find what the purpose of that port number is. Check out this link for port numbers used by Trojans: <http://www.simovits.com/trojans/trojans.html>.

26.6.5.6. An Example Inclusive Ruleset

The following non-NATed rule set is a complete inclusive type ruleset. You can not go wrong using this rule set for you own. Just comment out any pass rules for services you do not want. If you see messages in your log that you want to stop seeing just add a deny rule in the inbound section. You have to change the 'dc0' interface name in every rule to the interface name of the NIC that connects your system to the public Internet. For user ppp it would be 'tun0'.

You will see a pattern in the usage of these rules.

- All statements that are a request to start a session to the public Internet use keep-state.
- All the authorized services that originate from the public Internet have the limit option to stop flooding.
- All rules use in or out to clarify direction.
- All rules use via interface name to specify the interface the packet is traveling over.

The following rules go into `/etc/ipfw.rules`.

```
##### Start of IPFW rules file #####
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
pif="dc0"      # public interface name of NIC
               # facing the public Internet

#####
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN.
# Change xl0 to your LAN NIC interface name
#####
$cmd 00005 allow all from any to any via xl0

#####
# No restrictions on Loopback Interface
#####
$cmd 00010 allow all from any to any via lo0

#####
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
#####
$cmd 00015 check-state
```

```
#####
# Interface facing Public Internet (Outbound Section)
# Interrogate session start requests originating from behind the
# firewall on the private network or from this gateway server
# destined for the public Internet.
#####

# Allow out access to my ISP's Domain name server.
# x.x.x.x must be the IP address of your ISP's DNS
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
$cmd 00110 allow tcp from any to x.x.x.x 53 out via $pif setup keep-state
$cmd 00111 allow udp from any to x.x.x.x 53 out via $pif keep-state

# Allow out access to my ISP's DHCP server for cable/DSL configurations.
# This rule is not needed for .user ppp. connection to the public Internet.
# so you can delete this whole group.
# Use the following rule and check log for IP address.
# Then put IP address in commented out rule & delete first rule
$cmd 00120 allow log udp from any to any 67 out via $pif keep-state
#$cmd 00120 allow udp from any to x.x.x.x 67 out via $pif keep-state

# Allow out non-secure standard www function
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state

# Allow out secure www function https over TLS SSL
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state

# Allow out send & get email function
$cmd 00230 allow tcp from any to any 25 out via $pif setup keep-state
$cmd 00231 allow tcp from any to any 110 out via $pif setup keep-state

# Allow out FBSD (make install & CVSUP) functions
# Basically give user root "GOD" privileges.
$cmd 00240 allow tcp from me to any out via $pif setup keep-state uid root

# Allow out ping
$cmd 00250 allow icmp from any to any out via $pif keep-state

# Allow out Time
$cmd 00260 allow tcp from any to any 37 out via $pif setup keep-state

# Allow out nntp news (i.e. news groups)
$cmd 00270 allow tcp from any to any 119 out via $pif setup keep-state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
$cmd 00280 allow tcp from any to any 22 out via $pif setup keep-state

# Allow out whois
$cmd 00290 allow tcp from any to any 43 out via $pif setup keep-state

# deny and log everything else that's trying to get out.
```

```

# This rule enforces the block all by default logic.
$cmd 00299 deny log all from any to any out via $pif

#####
# Interface facing Public Internet (Inbound Section)
# Interrogate packets originating from the public Internet
# destined for this gateway server or the private network.
#####

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 00300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918 private IP
$cmd 00301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918 private IP
$cmd 00302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918 private IP
$cmd 00303 deny all from 127.0.0.0/8 to any in via $pif #loopback
$cmd 00304 deny all from 0.0.0.0/8 to any in via $pif #loopback
$cmd 00305 deny all from 169.254.0.0/16 to any in via $pif #DHCP auto-config
$cmd 00306 deny all from 192.0.2.0/24 to any in via $pif #reserved for docs
$cmd 00307 deny all from 204.152.64.0/23 to any in via $pif #Sun cluster interconnect
$cmd 00308 deny all from 224.0.0.0/3 to any in via $pif #Class D & E multicast

# Deny public pings
$cmd 00310 deny icmp from any to any in via $pif

# Deny ident
$cmd 00315 deny tcp from any to any 113 in via $pif

# Deny all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
$cmd 00320 deny tcp from any to any 137 in via $pif
$cmd 00321 deny tcp from any to any 138 in via $pif
$cmd 00322 deny tcp from any to any 139 in via $pif
$cmd 00323 deny tcp from any to any 81 in via $pif

# Deny any late arriving packets
$cmd 00330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 00332 deny tcp from any to any established in via $pif

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it's the only
# authorized source to send this packet type.
# Only necessary for cable or DSL configurations.
# This rule is not needed for .user ppp. type connection to
# the public Internet. This is the same IP address you captured
# and used in the outbound section.
$cmd 00360 allow udp from any to x.x.x.x 67 in via $pif keep-state

# Allow in standard www function because I have apache server
$cmd 00400 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow in secure FTP, Telnet, and SCP from public Internet

```

```

$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID & PW are passed over public
# Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
$cmd 00420 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Reject & Log all incoming connections from the outside
$cmd 00499 deny log all from any to any in via $pif

# Everything else is denied by default
# deny and log all packets that fell through to see what they are
$cmd 00999 deny log all from any to any
##### End of IPFW rules file #####

```

26.6.5.7. An Example NAT and Stateful Ruleset

There are some additional configuration statements that need to be enabled to activate the NAT function of IPFW. The kernel source needs 'option divert' statement added to the other IPFWALL statements compiled into a custom kernel.

In addition to the normal IPFW options in `/etc/rc.conf`, the following are needed.

```

natd_enable="YES"                # Enable NATD function
natd_interface="rl0"            # interface name of public Internet NIC
natd_flags="-dynamic -m"        # -m = preserve port numbers if possible

```

Utilizing stateful rules with divert natd rule (Network Address Translation) greatly complicates the rule set coding logic. The positioning of the check-state, and 'divert natd' rules in the rule set becomes very critical. This is no longer a simple fall-through logic flow. A new action type is used, called 'skipto'. To use the skipto command it is mandatory that you number each rule so you know exactly where the skipto rule number is you are really jumping to.

The following is an uncommented example of one coding method, selected here to explain the sequence of the packet flow through the rule sets.

The processing flow starts with the first rule from the top of the rule file and progress one rule at a time deeper into the file until the end is reached or the packet being tested to the selection criteria matches and the packet is released out of the firewall. It is important to take notice of the location of rule numbers 100, 101, 450, 500, and 510. These rules control the translation of the outbound and inbound packets so their entries in the keep-state dynamic table always register the private LAN IP address. Next notice that all the allow and deny rules specified the direction the packet is going (IE outbound or inbound) and the interface. Also notice that all the start outbound session requests all skipto rule 500 for the network address translation.

Lets say a LAN user uses their web browser to get a web page. Web pages use port 80 to communicate over. So the packet enters the firewall, It does not match 100 because it is headed out not in. It passes rule 101 because this is the first packet so it has not been posted to the keep-state dynamic table yet. The packet finally comes to rule 125 and matches. It is outbound through the NIC facing the public Internet. The packet still has its source IP address as a private LAN IP address. On the match to this rule, two actions take place. The keep-state option will post this rule into the keep-state dynamic rules table and the specified action is executed. The action is part of the info posted to the dynamic table. In this case it is "skipto rule 500". Rule 500 NATs the packet IP address and out it goes. Remember this, this is very important. This packet makes its way to the destination and returns and enters the top of the rule set.

This time it does match rule 100 and has its destination IP address mapped back to its corresponding LAN IP address. It then is processed by the check-state rule, it's found in the table as an existing session conversation and released to the LAN. It goes to the LAN PC that sent it and a new packet is sent requesting another segment of the data from the remote server. This time it gets checked by the check-state rule and its outbound entry is found, the associated action, 'skipto 500', is executed. The packet jumps to rule 500 gets NATed and released on its way out.

On the inbound side, everything coming in that is part of an existing session conversation is being automatically handled by the check-state rule and the properly placed divert natd rules. All we have to address is denying all the bad packets and only allowing in the authorized services. Let's say there is an Apache server running on the firewall box and we want people on the public Internet to be able to access the local web site. The new inbound start request packet matches rule 100 and its IP address is mapped to LAN IP for the firewall box. The packet is then matched against all the nasty things we want to check for and finally matches against rule 425. On a match two things occur. The packet rule is posted to the keep-state dynamic table but this time any new session requests originating from that source IP address is limited to 2. This defends against DoS attacks of service running on the specified port number. The action is allow so the packet is released to the LAN. On return the check-state rule recognizes the packet as belonging to an existing session conversation sends it to rule 500 for NATing and released to outbound interface.

Example Ruleset #1:

```
#!/bin/sh
cmd="ipfw -q add"
skip="skipto 500"
pif=rl0
ks="keep-state"
good_tcpo="22,25,37,43,53,80,443,110,119"

ipfw -q -f flush

$cmd 002 allow all from any to any via xl0 # exclude LAN traffic
$cmd 003 allow all from any to any via lo0 # exclude loopback traffic

$cmd 100 divert natd ip from any to any in via $pif
$cmd 101 check-state

# Authorized outbound packets
$cmd 120 $skip udp from any to xx.168.240.2 53 out via $pif $ks
$cmd 121 $skip udp from any to xx.168.240.5 53 out via $pif $ks
$cmd 125 $skip tcp from any to any $good_tcpo out via $pif setup $ks
$cmd 130 $skip icmp from any to any out via $pif $ks
$cmd 135 $skip udp from any to any 123 out via $pif $ks

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918 private IP
$cmd 301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918 private IP
$cmd 302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918 private IP
$cmd 303 deny all from 127.0.0.0/8 to any in via $pif #loopback
$cmd 304 deny all from 0.0.0.0/8 to any in via $pif #loopback
$cmd 305 deny all from 169.254.0.0/16 to any in via $pif #DHCP auto-config
$cmd 306 deny all from 192.0.2.0/24 to any in via $pif #reserved for docs
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif #Sun cluster
$cmd 308 deny all from 224.0.0.0/3 to any in via $pif #Class D & E multicast
```

```
# Authorized inbound packets
$cmd 400 allow udp from xx.70.207.54 to any 68 in $ks
$cmd 420 allow tcp from any to me 80 in via $pif setup limit src-addr 1

$cmd 450 deny log ip from any to any

# This is skipto location for outbound stateful rules
$cmd 500 divert natd ip from any to any out via $pif
$cmd 510 allow ip from any to any

##### end of rules #####
```

The following is pretty much the same as above, but uses a self documenting coding style full of description comments to help the inexperienced IPFW rule writer to better understand what the rules are doing.

Example Ruleset #2:

```
#!/bin/sh
##### Start of IPFW rules file #####
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
skip="skipto 800"
pif="rl0"      # public interface name of NIC
               # facing the public Internet

#####
# No restrictions on Inside LAN Interface for private network
# Change xl0 to your LAN NIC interface name
#####
$cmd 005 allow all from any to any via xl0

#####
# No restrictions on Loopback Interface
#####
$cmd 010 allow all from any to any via lo0

#####
# check if packet is inbound and nat address if it is
#####
$cmd 014 divert natd ip from any to any in via $pif

#####
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
#####
$cmd 015 check-state

#####
# Interface facing Public Internet (Outbound Section)
# Interrogate session start requests originating from behind the
```

```

# firewall on the private network or from this gateway server
# destine for the public Internet.
#####

# Allow out access to my ISP's Domain name server.
# x.x.x.x must be the IP address of your ISP's DNS
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
$cmd 020 $skip tcp from any to x.x.x.x 53 out via $pif setup keep-state

# Allow out access to my ISP's DHCP server for cable/DSL configurations.
$cmd 030 $skip udp from any to x.x.x.x 67 out via $pif keep-state

# Allow out non-secure standard www function
$cmd 040 $skip tcp from any to any 80 out via $pif setup keep-state

# Allow out secure www function https over TLS SSL
$cmd 050 $skip tcp from any to any 443 out via $pif setup keep-state

# Allow out send & get email function
$cmd 060 $skip tcp from any to any 25 out via $pif setup keep-state
$cmd 061 $skip tcp from any to any 110 out via $pif setup keep-state

# Allow out FreeBSD (make install & CVSUP) functions
# Basically give user root "GOD" privileges.
$cmd 070 $skip tcp from me to any out via $pif setup keep-state uid root

# Allow out ping
$cmd 080 $skip icmp from any to any out via $pif keep-state

# Allow out Time
$cmd 090 $skip tcp from any to any 37 out via $pif setup keep-state

# Allow out nntp news (i.e. news groups)
$cmd 100 $skip tcp from any to any 119 out via $pif setup keep-state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
$cmd 110 $skip tcp from any to any 22 out via $pif setup keep-state

# Allow out whois
$cmd 120 $skip tcp from any to any 43 out via $pif setup keep-state

# Allow ntp time server
$cmd 130 $skip udp from any to any 123 out via $pif keep-state

#####
# Interface facing Public Internet (Inbound Section)
# Interrogate packets originating from the public Internet
# destine for this gateway server or the private network.
#####

```

```

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918 private IP
$cmd 301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918 private IP
$cmd 302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918 private IP
$cmd 303 deny all from 127.0.0.0/8 to any in via $pif #loopback
$cmd 304 deny all from 0.0.0.0/8 to any in via $pif #loopback
$cmd 305 deny all from 169.254.0.0/16 to any in via $pif #DHCP auto-config
$cmd 306 deny all from 192.0.2.0/24 to any in via $pif #reserved for docs
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif #Sun cluster
$cmd 308 deny all from 224.0.0.0/3 to any in via $pif #Class D & E multicast

# Deny ident
$cmd 315 deny tcp from any to any 113 in via $pif

# Deny all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
$cmd 320 deny tcp from any to any 137 in via $pif
$cmd 321 deny tcp from any to any 138 in via $pif
$cmd 322 deny tcp from any to any 139 in via $pif
$cmd 323 deny tcp from any to any 81 in via $pif

# Deny any late arriving packets
$cmd 330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 332 deny tcp from any to any established in via $pif

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it's the only
# authorized source to send this packet type.
# Only necessary for cable or DSL configurations.
# This rule is not needed for 'user ppp' type connection to
# the public Internet. This is the same IP address you captured
# and used in the outbound section.
$cmd 360 allow udp from x.x.x.x to any 68 in via $pif keep-state

# Allow in standard www function because I have Apache server
$cmd 370 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow in secure FTP, Telnet, and SCP from public Internet
$cmd 380 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID & PW are passed over public
# Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
$cmd 390 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Reject & Log all unauthorized incoming connections from the public Internet
$cmd 400 deny log all from any to any in via $pif

# Reject & Log all unauthorized out going connections to the public Internet

```

```
$cmd 450 deny log all from any to any out via $pif

# This is skipto location for outbound stateful rules
$cmd 800 divert natd ip from any to any out via $pif
$cmd 801 allow ip from any to any

# Everything else is denied by default
# deny and log all packets that fell through to see what they are
$cmd 999 deny log all from any to any
##### End of IPFW rules file #####
```

Rozdział 27. Advanced Networking

27.1. Synopsis

This chapter will cover a number of advanced networking topics.

After reading this chapter, you will know:

- The basics of gateways and routes.
- How to set up IEEE 802.11 and Bluetooth® devices.
- How to make FreeBSD act as a bridge.
- How to set up network booting on a diskless machine.
- How to set up network address translation.
- How to connect two computers via PLIP.
- How to set up IPv6 on a FreeBSD machine.
- How to configure ATM.

Before reading this chapter, you should:

- Understand the basics of the `/etc/rc` scripts.
- Be familiar with basic network terminology.
- Know how to configure and install a new FreeBSD kernel (Rozdział 8).
- Know how to install additional third-party software (Rozdział 4).

27.2. Gateways and Routes

Contributed by Coranth Gryphon.

For one machine to be able to find another over a network, there must be a mechanism in place to describe how to get from one to the other. This is called *routing*. A „route” is a defined pair of addresses: a „destination” and a „gateway”. The pair indicates that if you are trying to get to this *destination*, communicate through this *gateway*. There are three types of destinations: individual hosts, subnets, and „default”. The „default route” is used if none of the other routes apply. We will talk a little bit more about default routes later on. There are also three types of gateways: individual hosts, interfaces (also called „links”), and Ethernet hardware addresses (MAC addresses).

27.2.1. An Example

To illustrate different aspects of routing, we will use the following example from `netstat`:

```
% netstat -r
Routing tables
```

Destination	Gateway	Flags	Refs	Use	Netif	Expire
-------------	---------	-------	------	-----	-------	--------

```

default        outside-gw      UGSc      37        418        ppp0
localhost      localhost      UH         0         181        lo0
test0          0:e0:b5:36:cf:4f UHLW       5        63288      ed0        77
10.20.30.255    link#1        UHLW       1         2421
example.com     link#1        UC         0          0
host1          0:e0:a8:37:8:1e UHLW       3         4601        lo0
host2          0:e0:a8:37:8:1e UHLW       0          5        lo0 =>
host2.example.com link#1        UC         0          0
224            link#1        UC         0          0

```

The first two lines specify the default route (which we will cover in the next section) and the `localhost` route.

The interface (Netif column) that this routing table specifies to use for `localhost` is `lo0`, also known as the loopback device. This says to keep all traffic for this destination internal, rather than sending it out over the LAN, since it will only end up back where it started.

The next thing that stands out are the addresses beginning with `0:e0:`. These are Ethernet hardware addresses, which are also known as MAC addresses. FreeBSD will automatically identify any hosts (`test0` in the example) on the local Ethernet and add a route for that host, directly to it over the Ethernet interface, `ed0`. There is also a timeout (Expire column) associated with this type of route, which is used if we fail to hear from the host in a specific amount of time. When this happens, the route to this host will be automatically deleted. These hosts are identified using a mechanism known as RIP (Routing Information Protocol), which figures out routes to local hosts based upon a shortest path determination.

FreeBSD will also add subnet routes for the local subnet (`10.20.30.255` is the broadcast address for the subnet `10.20.30`, and `example.com` is the domain name associated with that subnet). The designation `link#1` refers to the first Ethernet card in the machine. You will notice no additional interface is specified for those.

Both of these groups (local network hosts and local subnets) have their routes automatically configured by a daemon called **routed**. If this is not run, then only routes which are statically defined (i.e. entered explicitly) will exist.

The `host1` line refers to our host, which it knows by Ethernet address. Since we are the sending host, FreeBSD knows to use the loopback interface (`lo0`) rather than sending it out over the Ethernet interface.

The two `host2` lines are an example of what happens when we use an `ifconfig(8)` alias (see the section on Ethernet for reasons why we would do this). The `=>` symbol after the `lo0` interface says that not only are we using the loopback (since this address also refers to the local host), but specifically it is an alias. Such routes only show up on the host that supports the alias; all other hosts on the local network will simply have a `link#1` line for such routes.

The final line (destination subnet `224`) deals with multicasting, which will be covered in another section.

Finally, various attributes of each route can be seen in the `Flags` column. Below is a short table of some of these flags and their meanings:

U	Up: The route is active.
H	Host: The route destination is a single host.
G	Gateway: Send anything for this destination on to this remote system, which will figure out from there where to send it.
S	Static: This route was configured manually, not automatically generated by the system.
C	Clone: Generates a new route based upon this route for machines we connect to. This type of route is normally used for local networks.

- W WasCloned: Indicated a route that was auto-configured based upon a local area network (Clone) route.
- L Link: Route involves references to Ethernet hardware.

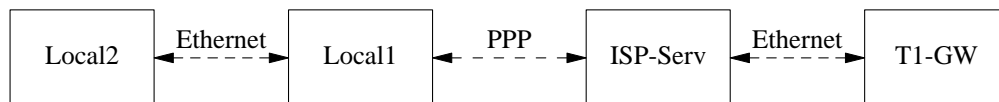
27.2.2. Default Routes

When the local system needs to make a connection to a remote host, it checks the routing table to determine if a known path exists. If the remote host falls into a subnet that we know how to reach (Cloned routes), then the system checks to see if it can connect along that interface.

If all known paths fail, the system has one last option: the „default” route. This route is a special type of gateway route (usually the only one present in the system), and is always marked with a `c` in the flags field. For hosts on a local area network, this gateway is set to whatever machine has a direct connection to the outside world (whether via PPP link, DSL, cable modem, T1, or another network interface).

If you are configuring the default route for a machine which itself is functioning as the gateway to the outside world, then the default route will be the gateway machine at your Internet Service Provider’s (ISP) site.

Let us look at an example of default routes. This is a common configuration:



The hosts `Local1` and `Local2` are at your site. `Local1` is connected to an ISP via a dial up PPP connection. This PPP server computer is connected through a local area network to another gateway computer through an external interface to the ISP’s Internet feed.

The default routes for each of your machines will be:

Host	Default Gateway	Interface
Local2	Local1	Ethernet
Local1	T1-GW	PPP

A common question is „Why (or how) would we set the `T1-GW` to be the default gateway for `Local1`, rather than the ISP server it is connected to?”.

Remember, since the PPP interface is using an address on the ISP’s local network for your side of the connection, routes for any other machines on the ISP’s local network will be automatically generated. Hence, you will already know how to reach the `T1-GW` machine, so there is no need for the intermediate step of sending traffic to the ISP server.

It is common to use the address `x.x.x.1` as the gateway address for your local network. So (using the same example), if your local class-C address space was `10.20.30` and your ISP was using `10.9.9` then the default routes would be:

Host	Default Route
Local2 (10.20.30.2)	Local1 (10.20.30.1)
Local1 (10.20.30.1, 10.9.9.30)	T1-GW (10.9.9.1)

You can easily define the default route via the `/etc/rc.conf` file. In our example, on the `Local2` machine, we added the following line in `/etc/rc.conf`:

```
defaultrouter="10.20.30.1"
```

It is also possible to do it directly from the command line with the `route(8)` command:

```
# route add default 10.20.30.1
```

For more information on manual manipulation of network routing tables, consult `route(8)` manual page.

27.2.3. Dual Homed Hosts

There is one other type of configuration that we should cover, and that is a host that sits on two different networks. Technically, any machine functioning as a gateway (in the example above, using a PPP connection) counts as a dual-homed host. But the term is really only used to refer to a machine that sits on two local-area networks.

In one case, the machine has two Ethernet cards, each having an address on the separate subnets. Alternately, the machine may only have one Ethernet card, and be using `ifconfig(8)` aliasing. The former is used if two physically separate Ethernet networks are in use, the latter if there is one physical network segment, but two logically separate subnets.

Either way, routing tables are set up so that each subnet knows that this machine is the defined gateway (inbound route) to the other subnet. This configuration, with the machine acting as a router between the two subnets, is often used when we need to implement packet filtering or firewall security in either or both directions.

If you want this machine to actually forward packets between the two interfaces, you need to tell FreeBSD to enable this ability. See the next section for more details on how to do this.

27.2.4. Building a Router

A network router is simply a system that forwards packets from one interface to another. Internet standards and good engineering practice prevent the FreeBSD Project from enabling this by default in FreeBSD. You can enable this feature by changing the following variable to `YES` in `rc.conf(5)`:

```
gateway_enable=YES          # Set to YES if this host will be a gateway
```

This option will set the `sysctl(8)` variable `net.inet.ip.forwarding` to 1. If you should need to stop routing temporarily, you can reset this to 0 temporarily.

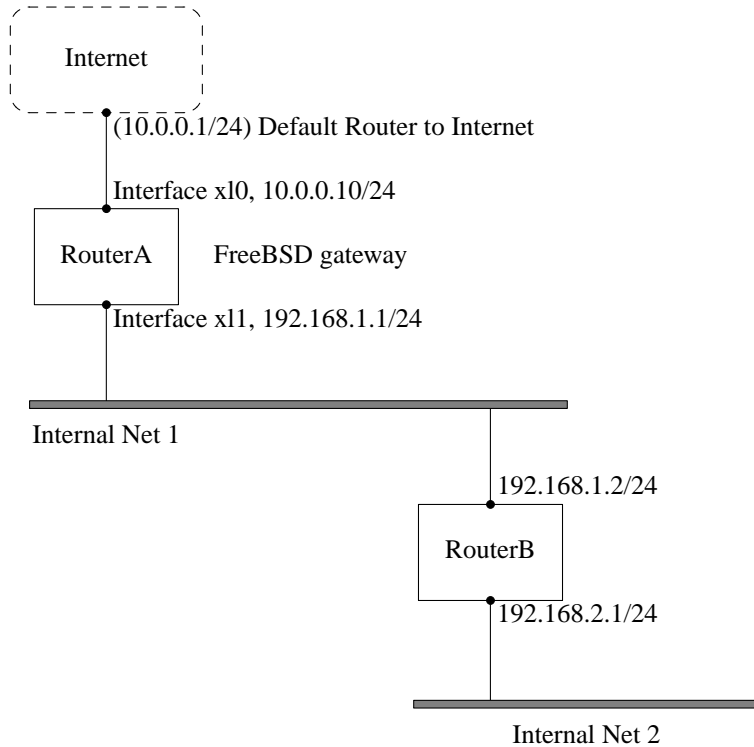
Your new router will need routes to know where to send the traffic. If your network is simple enough you can use static routes. FreeBSD also comes with the standard BSD routing daemon `routed(8)`, which speaks RIP (both version 1 and version 2) and IRDP. Support for BGP v4, OSPF v2, and other sophisticated routing protocols is available with the `net/zebra` package. Commercial products such as **GateD** are also available for more complex network routing solutions.

27.2.5. Setting Up Static Routes

Contributed by Al Hoang.

27.2.5.1. Manual Configuration

Let us assume we have a network as follows:



In this scenario, RouterA is our FreeBSD machine that is acting as a router to the rest of the Internet. It has a default route set to 10.0.0.1 which allows it to connect with the outside world. We will assume that RouterB is already configured properly and knows how to get wherever it needs to go. (This is simple in this picture. Just add a default route on RouterB using 192.168.1.1 as the gateway.)

If we look at the routing table for RouterA we would see something like the following:

```
% netstat -nr
Routing tables
```

```
Internet:
Destination      Gateway          Flags    Refs      Use  Netif  Expire
default          10.0.0.1        UGS      0         49378   xl0
127.0.0.1        127.0.0.1       UH        0          6     lo0
10.0.0/24        link#1          UC        0          0     xl0
192.168.1/24     link#2          UC        0          0     xl1
```

With the current routing table RouterA will not be able to reach our Internal Net 2. It does not have a route for 192.168.2.0/24. One way to alleviate this is to manually add the route. The following command would add the Internal Net 2 network to RouterA's routing table using 192.168.1.2 as the next hop:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Now RouterA can reach any hosts on the 192.168.2.0/24 network.

27.2.5.2. Persistent Configuration

The above example is perfect for configuring a static route on a running system. However, one problem is that the routing information will not persist if you reboot your FreeBSD machine. The way to handle the addition of a static route is to put it in your `/etc/rc.conf` file:

```
# Add Internal Net 2 as a static route
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

The `static_routes` configuration variable is a list of strings separated by a space. Each string references to a route name. In our above example we only have one string in `static_routes`. This string is `internalnet2`. We then add a configuration variable called `route_internalnet2` where we put all of the configuration parameters we would give to the `route(8)` command. For our example above we would have used the command:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

so we need `"-net 192.168.2.0/24 192.168.1.2"`.

As said above, we can have more than one string in `static_routes`. This allows us to create multiple static routes. The following lines shows an example of adding static routes for the 192.168.0.0/24 and 192.168.1.0/24 networks on an imaginary router:

```
static_routes="net1 net2"
route_net1="-net 192.168.0.0/24 192.168.0.1"
route_net2="-net 192.168.1.0/24 192.168.1.1"
```

27.2.6. Routing Propagation

We have already talked about how we define our routes to the outside world, but not about how the outside world finds us.

We already know that routing tables can be set up so that all traffic for a particular address space (in our examples, a class-C subnet) can be sent to a particular host on that network, which will forward the packets inbound.

When you get an address space assigned to your site, your service provider will set up their routing tables so that all traffic for your subnet will be sent down your PPP link to your site. But how do sites across the country know to send to your ISP?

There is a system (much like the distributed DNS information) that keeps track of all assigned address-spaces, and defines their point of connection to the Internet Backbone. The „Backbone” are the main trunk lines that carry Internet traffic across the country, and around the world. Each backbone machine has a copy of a master set of tables, which direct traffic for a particular network to a specific backbone carrier, and from there down the chain of service providers until it reaches your network.

It is the task of your service provider to advertise to the backbone sites that they are the point of connection (and thus the path inward) for your site. This is known as route propagation.

27.2.7. Troubleshooting

Sometimes, there is a problem with routing propagation, and some sites are unable to connect to you. Perhaps the most useful command for trying to figure out where routing is breaking down is the `tracert(8)` command. It is equally useful if you cannot seem to make a connection to a remote machine (i.e. `ping(8)` fails).

The `tracert(8)` command is run with the name of the remote host you are trying to connect to. It will show the gateway hosts along the path of the attempt, eventually either reaching the target host, or terminating because of a lack of connection.

For more information, see the manual page for `tracert(8)`.

27.2.8. Multicast Routing

FreeBSD supports both multicast applications and multicast routing natively. Multicast applications do not require any special configuration of FreeBSD; applications will generally run out of the box. Multicast routing requires that support be compiled into the kernel:

```
options MROUTING
```

In addition, the multicast routing daemon, `mROUTED(8)` must be configured to set up tunnels and DVMRP via `/etc/mROUTED.conf`. More details on multicast configuration may be found in the manual page for `mROUTED(8)`.

27.3. Wireless Networking

Loader, Marc Fonvieille, i Murray Stokely.

27.3.1. Wireless Networking Basics

Most wireless networks are based on the IEEE 802.11 standards. A basic wireless network consists of multiple stations communicating with radios that broadcast in either the 2.4GHz or 5GHz band (though this varies according to the locale and is also changing to enable communication in the 2.3GHz and 4.9GHz ranges).

802.11 networks are organized in two ways: in *infrastructure mode* one station acts as a master with all the other stations associating to it; the network is known as a BSS and the master station is termed an access point (AP). In a BSS all communication passes through the AP; even when one station wants to communicate with another wireless station messages must go through the AP. In the second form of network there is no master and stations communicate directly. This form of network is termed an IBSS and is commonly known as an *ad-hoc network*.

802.11 networks were first deployed in the 2.4GHz band using protocols defined by the IEEE 802.11 and 802.11b standard. These specifications include the operating frequencies, MAC layer characteristics including framing and transmission rates (communication can be done at various rates). Later the 802.11a standard defined operation in the 5GHz band, including different signalling mechanisms and higher transmission rates. Still later the 802.11g standard was defined to enable use of 802.11a signalling and transmission mechanisms in the 2.4GHz band in such a way as to be backwards compatible with 802.11b networks.

Separate from the underlying transmission techniques 802.11 networks have a variety of security mechanisms. The original 802.11 specifications defined a simple security protocol called WEP. This protocol uses a fixed pre-shared key and the RC4 cryptographic cipher to encode data transmitted on a network. Stations must all agree on the fixed

key in order to communicate. This scheme was shown to be easily broken and is now rarely used except to discourage transient users from joining networks. Current security practice is given by the IEEE 802.11i specification that defines new cryptographic ciphers and an additional protocol to authenticate stations to an access point and exchange keys for doing data communication. Further, cryptographic keys are periodically refreshed and there are mechanisms for detecting intrusion attempts (and for countering intrusion attempts). Another security protocol specification commonly used in wireless networks is termed WPA. This was a precursor to 802.11i defined by an industry group as an interim measure while waiting for 802.11i to be ratified. WPA specifies a subset of the requirements found in 802.11i and is designed for implementation on legacy hardware. Specifically WPA requires only the TKIP cipher that is derived from the original WEP cipher. 802.11i permits use of TKIP but also requires support for a stronger cipher, AES-CCM, for encrypting data. (The AES cipher was not required in WPA because it was deemed too computationally costly to be implemented on legacy hardware.)

Other than the above protocol standards the other important standard to be aware of is 802.11e. This defines protocols for deploying multi-media applications such as streaming video and voice over IP (VoIP) in an 802.11 network. Like 802.11i, 802.11e also has a precursor specification termed WME (later renamed WMM) that has been defined by an industry group as a subset of 802.11e that can be deployed now to enable multi-media applications while waiting for the final ratification of 802.11e. The most important thing to know about 802.11e and WME/WMM is that it enables prioritized traffic use of a wireless network through Quality of Service (QoS) protocols and enhanced media access protocols. Proper implementation of these protocols enable high speed bursting of data and prioritized traffic flow.

Since the 6.0 version, FreeBSD supports networks that operate using 802.11a, 802.11b, and 802.11g. The WPA and 802.11i security protocols are likewise supported (in conjunction with any of 11a, 11b, and 11g) and QoS and traffic prioritization required by the WME/WMM protocols are supported for a limited set of wireless devices.

27.3.2. Basic Setup

27.3.2.1. Kernel Configuration

To use wireless networking you need a wireless networking card and to configure the kernel with the appropriate wireless networking support. The latter is separated into multiple modules so that you only need to configure the software you are actually going to use.

The first thing you need is a wireless device. The most commonly used devices are those that use parts made by Atheros. These devices are supported by the ath(4) driver and require the following line to be added to the `/boot/loader.conf` file:

```
if_ath_load="YES"
```

The Atheros driver is split up into three separate pieces: the driver proper (ath(4)), the hardware support layer that handles chip-specific functions (ath_hal(4)), and an algorithm for selecting which of several possible rates for transmitting frames (ath_rate_sample here). When you load this support as modules these dependencies are automatically handled for you. If instead of an Atheros device you had another device you would select the module for that device; e.g.:

```
if_wi_load="YES"
```

for devices based on the Intersil Prism parts (wi(4) driver).

Notatka: In the rest of this document, we will use an ath(4) device, the device name in the examples must be changed according to your configuration. A list of available wireless drivers can be found at the beginning of the

wlan(4) manual page. If a native FreeBSD driver for your wireless device does not exist, it may be possible to directly use the Windows driver with the help of the NDIS driver wrapper.

With a device driver configured you need to also bring in the 802.11 networking support required by the driver. For the ath(4) driver this is at least the wlan(4) module; this module is automatically loaded with the wireless device driver. With that you will need the modules that implement cryptographic support for the security protocols you intend to use. These are intended to be dynamically loaded on demand by the wlan(4) module but for now they must be manually configured. The following modules are available: wlan_wep(4), wlan_ccmp(4) and wlan_tkip(4). Both wlan_ccmp(4) and wlan_tkip(4) drivers are only needed if you intend to use the WPA and/or 802.11i security protocols. If your network is to run totally open (i.e., with no encryption) then you do not even need the wlan_wep(4) support. To load these modules at boot time, add the following lines to `/boot/loader.conf`:

```
wlan_wep_load="YES"
wlan_ccmp_load="YES"
wlan_tkip_load="YES"
```

With this information in the system bootstrap configuration file (i.e., `/boot/loader.conf`), you have to reboot your FreeBSD box. If you do not want to reboot your machine for the moment, you can just load the modules by hand using `kldload(8)`.

Notatka: If you do not want to use modules, it is possible to compile these drivers into the kernel by adding the following lines to your kernel configuration file:

```
device ath          # Atheros IEEE 802.11 wireless network driver
device ath_hal      # Atheros Hardware Access Layer
device ath_rate_sample # John Bicket's SampleRate control algorithm.
device wlan         # 802.11 support (Required)
device wlan_wep     # WEP crypto support for 802.11 devices
device wlan_ccmp     # AES-CCMP crypto support for 802.11 devices
device wlan_tkip     # TKIP and Michael crypto support for 802.11 devices
```

With this information in the kernel configuration file, recompile the kernel and reboot your FreeBSD machine.

When the system is up, we could find some information about the wireless device in the boot messages, like this:

```
ath0: <Atheros 5212> mem 0xff9f0000-0xff9fffff irq 17 at device 2.0 on pci2
ath0: Ethernet address: 00:11:95:d5:43:62
ath0: mac 7.9 phy 4.5 radio 5.6
```

27.3.3. Infrastructure Mode

The infrastructure mode or BSS mode is the mode that is typically used. In this mode, a number of wireless access points are connected to a wired network. Each wireless network has its own name, this name is called the SSID of the network. Wireless clients connect to the wireless access points.

27.3.3.1. FreeBSD Clients

27.3.3.1.1. How to Find Access Points

To scan for networks, use the `ifconfig` command. This request may take a few moments to complete as it requires that the system switches to each available wireless frequency and probes for available access points. Only the super-user can initiate such a scan:

```
# ifconfig ath0 up scan
SSID          BSSID          CHAN  RATE  S:N    INT  CAPS
dlinkap       00:13:46:49:41:76    6    54M  29:0   100  EPS   WPA WME
freebsdap     00:11:95:c3:0d:ac    1    54M  22:0   100  EPS   WPA
```

Notatka: You must mark the interface `up` before you can scan. Subsequent scan requests do not require you to mark the interface `up` again.

The output of a scan request lists each BSS/IBSS network found. Beside the name of the network, `SSID`, we find the `BSSID` which is the MAC address of the access point. The `CAPS` field identifies the type of each network and the capabilities of the stations operating there:

E

Extended Service Set (ESS). Indicates that the station is part of an infrastructure network (in contrast to an IBSS/ad-hoc network).

I

IBSS/ad-hoc network. Indicates that the station is part of an ad-hoc network (in contrast to an ESS network).

P

Privacy. Data confidentiality is required for all data frames exchanged within the BSS. This means that this BSS requires the station to use cryptographic means such as WEP, TKIP or AES-CCMP to encrypt/decrypt data frames being exchanged with others.

S

Short Preamble. Indicates that the network is using short preambles (defined in 802.11b High Rate/DSSS PHY, short preamble utilizes a 56 bit sync field in contrast to a 128 bit field used in long preamble mode).

s

Short slot time. Indicates that the 802.11g network is using a short slot time because there are no legacy (802.11b) stations present.

One can also display the current list of known networks with:

```
# ifconfig ath0 list scan
```

This information may be updated automatically by the adapter or manually with a `scan` request. Old data is automatically removed from the cache, so over time this list may shrink unless more scans are done.

27.3.3.1.2. Basic Settings

This section provides a simple example of how to make the wireless network adapter work in FreeBSD without encryption. After you are familiar with these concepts, we strongly recommend using WPA to set up your wireless network.

There are three basic steps to configure a wireless network: selecting an access point, authenticating your station, and configuring an IP address. The following sections discuss each step.

27.3.3.1.2.1. Selecting an Access Point

Most of time it is sufficient to let the system choose an access point using the builtin heuristics. This is the default behaviour when you mark an interface up or otherwise configure an interface by listing it in `/etc/rc.conf`, e.g.:

```
ifconfig_ath0="DHCP"
```

If there are multiple access points and you want to select a specific one, you can select it by its SSID:

```
ifconfig_ath0="ssid your_ssid_here DHCP"
```

In an environment where there are multiple access points with the same SSID (often done to simplify roaming) it may be necessary to associate to one specific device. In this case you can also specify the BSSID of the access point (you can also leave off the SSID):

```
ifconfig_ath0="ssid your_ssid_here bssid xx:xx:xx:xx:xx:xx DHCP"
```

There are other ways to constrain the choice of an access point such as limiting the set of frequencies the system will scan on. This may be useful if you have a multi-band wireless card as scanning all the possible channels can be time-consuming. To limit operation to a specific band you can use the `mode` parameter; e.g.:

```
ifconfig_ath0="mode 11g ssid your_ssid_here DHCP"
```

will force the card to operate in 802.11g which is defined only for 2.4GHz frequencies so any 5GHz channels will not be considered. Other ways to do this are the `channel` parameter, to lock operation to one specific frequency, and the `chanlist` parameter, to specify a list of channels for scanning. More information about these parameters can be found in the `ifconfig(8)` manual page.

27.3.3.1.2.2. Authentication

Once you have selected an access point your station needs to authenticate before it can pass data. Authentication can happen in several ways. The most common scheme used is termed open authentication and allows any station to join the network and communicate. This is the authentication you should use for test purpose the first time you set up a wireless network. Other schemes require cryptographic handshakes be completed before data traffic can flow; either using pre-shared keys or secrets, or more complex schemes that involve backend services such as RADIUS. Most users will use open authentication which is the default setting. Next most common setup is WPA-PSK, also known as WPA Personal, which is described below.

Notatka: If you have an Apple® AirPort® Extreme base station for an access point you may need to configure shared-key authentication together with a WEP key. This can be done in the `/etc/rc.conf` file or using the `wpa_supplicant(8)` program. If you have a single AirPort base station you can setup access with something like:

```
ifconfig_ath0="authmode shared wepmode on weptxkey 1 wepkey 01234567 DHCP"
```

In general shared key authentication is to be avoided because it uses the WEP key material in a highly-constrained manner making it even easier to crack the key. If WEP must be used (e.g., for compatibility with legacy devices) it is better to use WEP with `open` authentication. More information regarding WEP can be found in the Sekcja 27.3.3.1.4.

27.3.3.1.2.3. Getting an IP Address with DHCP

Once you have selected an access point and set the authentication parameters, you will have to get an IP address to communicate. Most of time you will obtain your wireless IP address via DHCP. To achieve that, simply edit `/etc/rc.conf` and add DHCP to the configuration for your device as shown in various examples above:

```
ifconfig_ath0="DHCP"
```

At this point, you are ready to bring up the wireless interface:

```
# /etc/rc.d/netif start
```

Once the interface is running, use `ifconfig` to see the status of the interface `ath0`:

```
# ifconfig ath0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid 0x1
    inet 192.168.1.100 netmask 0xffffffff00 broadcast 192.168.1.255
    ether 00:11:95:d5:43:62
    media: IEEE 802.11 Wireless Ethernet autoselect (OFDM/54Mbps)
    status: associated
    ssid dlinkap channel 6 bssid 00:13:46:49:41:76
    authmode OPEN privacy OFF txpowmax 36 protmode CTS bintval 100
```

The `status: associated` means you are connected to the wireless network (to the `dlinkap` network in our case). The `bssid 00:13:46:49:41:76` part is the MAC address of your access point; the `authmode` line informs you that the communication is not encrypted (`OPEN`).

27.3.3.1.2.4. Static IP Address

In the case you cannot obtain an IP address from a DHCP server, you can set a fixed IP address. Replace the `DHCP` keyword shown above with the address information. Be sure to retain any other parameters you have set up for selecting an access point:

```
ifconfig_ath0="inet 192.168.1.100 netmask 255.255.255.0 ssid your_ssid_here"
```

27.3.3.1.3. WPA

WPA (Wi-Fi Protected Access) is a security protocol used together with 802.11 networks to address the lack of proper authentication and the weakness of WEP. WPA leverages the 802.1X authentication protocol and uses one of several ciphers instead of WEP for data integrity. The only cipher required by WPA is TKIP (Temporary Key Integrity Protocol) which is a cipher that extends the basic RC4 cipher used by WEP by adding integrity checking,

tamper detection, and measures for responding to any detected intrusions. TKIP is designed to work on legacy hardware with only software modification; it represents a compromise that improves security but is still not entirely immune to attack. WPA also specifies the AES-CCMP cipher as an alternative to TKIP and that is preferred when possible; for this specification the term WPA2 (or RSN) is commonly used.

WPA defines authentication and encryption protocols. Authentication is most commonly done using one of two techniques: by 802.1X and a backend authentication service such as RADIUS, or by a minimal handshake between the station and the access point using a pre-shared secret. The former is commonly termed WPA Enterprise with the latter known as WPA Personal. Since most people will not set up a RADIUS backend server for wireless network, WPA-PSK is by far the most commonly encountered configuration for WPA.

The control of the wireless connection and the authentication (key negotiation or authentication with a server) is done with the `wpa_supplicant(8)` utility. This program requires a configuration file, `/etc/wpa_supplicant.conf`, to run. More information regarding this file can be found in the `wpa_supplicant.conf(5)` manual page.

27.3.3.1.3.1. WPA-PSK

WPA-PSK also known as WPA-Personal is based on a pre-shared key (PSK) generated from a given password and that will be used as the master key in the wireless network. This means every wireless user will share the same key. WPA-PSK is intended for small networks where the use of an authentication server is not possible or desired.

Ostrzeżenie Always use strong passwords that are sufficiently long and made from a rich alphabet so they will not be guessed and/or attacked.

The first step is the configuration of the `/etc/wpa_supplicant.conf` file with the SSID and the pre-shared key of your network:

```
network={
    ssid="freebsdap"
    psk="freebsdmail"
}
```

Then, in `/etc/rc.conf`, we indicate that the wireless device configuration will be done with WPA and the IP address will be obtained with DHCP:

```
ifconfig_ath0="WPA DHCP"
```

Then, we can bring up the interface:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPDISCOVER on ath0 to 255.255.255.255 port 67 interval 5
DHCPDISCOVER on ath0 to 255.255.255.255 port 67 interval 6
DHCPOFFER from 192.168.0.1
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid 0x1
    inet 192.168.0.254 netmask 0xfffff00 broadcast 192.168.0.255
    ether 00:11:95:d5:43:62
```

```
media: IEEE 802.11 Wireless Ethernet autoselect (OFDM/36Mbps)
status: associated
ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
authmode WPA privacy ON deftxkey UNDEF TKIP 2:128-bit txpowmax 36
protmode CTS roaming MANUAL bintval 100
```

Or you can try to configure it manually using the same `/etc/wpa_supplicant.conf` above, and run:

```
# wpa_supplicant -i ath0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:11:95:c3:0d:ac (SSID='freebsdap' freq=2412 MHz)
Associated with 00:11:95:c3:0d:ac
WPA: Key negotiation completed with 00:11:95:c3:0d:ac [PTK=TKIP GTK=TKIP]
```

The next operation is the launch of the `dhclient` command to get the IP address from the DHCP server:

```
# dhclient ath0
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
# ifconfig ath0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid 0x1
    inet 192.168.0.254 netmask 0xffffffff broadcast 192.168.0.255
    ether 00:11:95:d5:43:62
    media: IEEE 802.11 Wireless Ethernet autoselect (OFDM/48Mbps)
    status: associated
    ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
    authmode WPA privacy ON deftxkey UNDEF TKIP 2:128-bit txpowmax 36
    protmode CTS roaming MANUAL bintval 100
```

Notatka: If the `/etc/rc.conf` is set up with the line `ifconfig_ath0="DHCP"` then it is no need to run the `dhclient` command manually, `dhclient` will be launched after `wpa_supplicant` plumbs the keys.

In the case where the use of DHCP is not possible, you can set a static IP address after `wpa_supplicant` has authenticated the station:

```
# ifconfig ath0 inet 192.168.0.100 netmask 255.255.255.0
# ifconfig ath0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid 0x1
    inet 192.168.0.100 netmask 0xffffffff broadcast 192.168.0.255
    ether 00:11:95:d5:43:62
    media: IEEE 802.11 Wireless Ethernet autoselect (OFDM/36Mbps)
    status: associated
    ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
    authmode WPA privacy ON deftxkey UNDEF TKIP 2:128-bit txpowmax 36
    protmode CTS roaming MANUAL bintval 100
```

When DHCP is not used, you also have to manually set up the default gateway and the nameserver:

```
# route add default your_default_router
```

```
# echo "nameserver your_DNS_server" >> /etc/resolv.conf
```

27.3.3.1.3.2. WPA with EAP-TLS

The second way to use WPA is with an 802.1X backend authentication server, in this case WPA is called WPA-Enterprise to make difference with the less secure WPA-Personal with its pre-shared key. The authentication in WPA-Enterprise is based on EAP (Extensible Authentication Protocol).

EAP does not come with an encryption method, it was decided to embed EAP inside an encrypted tunnel. Many types of EAP authentication methods have been designed, the most common methods are EAP-TLS, EAP-TTLS and EAP-PEAP.

EAP-TLS (EAP with Transport Layer Security) is a very well-supported authentication protocol in the wireless world since it was the first EAP method to be certified by the Wi-Fi alliance (<http://www.wi-fi.org/>). EAP-TLS will require three certificates to run: the CA certificate (installed on all machines), the server certificate for your authentication server, and one client certificate for each wireless client. In this EAP method, both authentication server and wireless client authenticate each other in presenting their respective certificates, and they verify that these certificates were signed by your organization's certificate authority (CA).

As previously, the configuration is done via `/etc/wpa_supplicant.conf`:

```
network={
    ssid="freebsdap" ❶
    proto=RSN ❷
    key_mgmt=WPA-EAP ❸
    eap=TLS ❹
    identity="loader" ❺
    ca_cert="/etc/certs/cacert.pem" ❻
    client_cert="/etc/certs/clientcert.pem" ❼
    private_key="/etc/certs/clientkey.pem" ❽
    private_key_passwd="freebsdmailclient" ❾
}
```

- ❶ This field indicates the network name (SSID).
- ❷ Here, we use RSN (IEEE 802.11i) protocol, i.e., WPA2.
- ❸ The `key_mgmt` line refers to the key management protocol we use. In our case it is WPA using EAP authentication: `WPA-EAP`.
- ❹ In this field, we mention the EAP method for our connection.
- ❺ The `identity` field contains the identity string for EAP.
- ❻ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificat.
- ❼ The `client_cert` line gives the pathname to the client certificate file. This certificate is unique to each wireless client of the network.
- ❽ The `private_key` field is the pathname to the client certificate private key file.
- ❾ The `private_key` field contains the passphrase for the private key.

Then add the following line to `/etc/rc.conf`:

```
ifconfig_ath0="WPA DHCP"
```

The next step is to bring up the interface with the help of the `rc.d` facility:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid 0x1
    inet 192.168.0.254 netmask 0xffffffff00 broadcast 192.168.0.255
    ether 00:11:95:d5:43:62
    media: IEEE 802.11 Wireless Ethernet autoselect (DS/11Mbps)
    status: associated
    ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
    authmode WPA2/802.11i privacy ON deftxkey UNDEF TKIP 2:128-bit
    txpowmax 36 protmode CTS roaming MANUAL bintval 100
```

As previously shown, it is also possible to bring up the interface manually with both `wpa_supplicant` and `ifconfig` commands.

27.3.3.1.3.3. WPA with EAP-TTLS

With EAP-TLS both the authentication server and the client need a certificate, with EAP-TTLS (EAP-Tunneled Transport Layer Security) a client certificate is optional. This method is close to what some secure web sites do, where the web server can create a secure SSL tunnel even if the visitors do not have client-side certificates. EAP-TTLS will use the encrypted TLS tunnel for safe transport of the authentication data.

The configuration is done via the `/etc/wpa_supplicant.conf` file:

```
network={
    ssid="freebsdap"
    proto=RSN
    key_mgmt=WPA-EAP
    eap=TTLS ❶
    identity="test" ❷
    password="test" ❸
    ca_cert="/etc/certs/cacert.pem" ❹
    phase2="auth=MD5" ❺
}
```

- ❶ In this field, we mention the EAP method for our connection.
- ❷ The `identity` field contains the identity string for EAP authentication inside the encrypted TLS tunnel.
- ❸ The `password` field contains the passphrase for the EAP authentication.
- ❹ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificate.

- ⑤ In this field, we mention the authentication method used in the encrypted TLS tunnel. In our case, EAP with MD5-Challenge has been used. The „inner authentication” phase is often called „phase2”.

You also have to add the following line to `/etc/rc.conf`:

```
ifconfig_ath0="WPA DHCP"
```

The next step is to bring up the interface:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid 0x1
    inet 192.168.0.254 netmask 0xffffffff broadcast 192.168.0.255
    ether 00:11:95:d5:43:62
    media: IEEE 802.11 Wireless Ethernet autoselect (DS/11Mbps)
    status: associated
    ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
    authmode WPA2/802.11i privacy ON deftxkey UNDEF TKIP 2:128-bit
    txpowmax 36 protmode CTS roaming MANUAL bintval 100
```

27.3.3.1.3.4. WPA with EAP-PEAP

PEAP (Protected EAP) has been designed as an alternative to EAP-TTLS. There are two types of PEAP methods, the most common one is PEAPv0/EAP-MSCHAPv2. In the rest of this document, we will use the PEAP term to refer to that EAP method. PEAP is the most used EAP standard after EAP-TLS, in other words if you have a network with mixed OSes, PEAP should be the most supported standard after EAP-TLS.

PEAP is similar to EAP-TTLS: it uses a server-side certificate to authenticate clients by creating an encrypted TLS tunnel between the client and the authentication server, which protects the ensuing exchange of authentication information. In terms of security the difference between EAP-TTLS and PEAP is that PEAP authentication broadcasts the username in clear, only the password is sent in the encrypted TLS tunnel. EAP-TTLS will use the TLS tunnel for both username and password.

We have to edit the `/etc/wpa_supplicant.conf` file and add the EAP-PEAP related settings:

```
network={
    ssid="freebsdap"
    proto=RSN
    key_mgmt=WPA-EAP
    eap=PEAP ①
    identity="test" ②
    password="test" ③
    ca_cert="/etc/certs/cacert.pem" ④
    phase1="peaplabel=0" ⑤
    phase2="auth=MSCHAPV2" ⑥
}
```

- ❶ In this field, we mention the EAP method for our connection.
- ❷ The `identity` field contains the identity string for EAP authentication inside the encrypted TLS tunnel.
- ❸ The `password` field contains the passphrase for the EAP authentication.
- ❹ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificat.
- ❺ This field contains the parameters for the first phase of the authentication (the TLS tunnel). According to the authentication server used, you will have to specify a specific label for the authentication. Most of time, the label will be „client EAP encryption” which is set by using `peaplabel=0`. More information can be found in the `wpa_supplicant.conf(5)` manual page.
- ❻ In this field, we mention the authentication protocol used in the encrypted TLS tunnel. In the case of PEAP, it is `auth=MSCHAPV2`.

The following must be added to `/etc/rc.conf`:

```
ifconfig_ath0="WPA DHCP"
```

Then, we can bring up the interface:

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPREQUEST on ath0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid 0x1
    inet 192.168.0.254 netmask 0xffffffff broadcast 192.168.0.255
    ether 00:11:95:d5:43:62
    media: IEEE 802.11 Wireless Ethernet autoselect (DS/11Mbps)
    status: associated
    ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
    authmode WPA2/802.11i privacy ON deftxkey UNDEF TKIP 2:128-bit
    txpowmax 36 protmode CTS roaming MANUAL bintval 100
```

27.3.3.1.4. WEP

WEP (Wired Equivalent Privacy) is part of the original 802.11 standard. There is no authentication mechanism, only a weak form of access control, and it is easily to be cracked.

WEP can be set up with `ifconfig`:

```
# ifconfig ath0 inet 192.168.1.100 netmask 255.255.255.0 ssid my_net \
    wepmode on weptxkey 3 wepkey 3:0x3456789012
```

- The `weptxkey` means which WEP key will be used in the transmission. Here we used the third key. This must match the setting in the access point.

- The `wepkey` means setting the selected WEP key. It should in the format `index:key`, if the index is not given, key 1 is set. That is to say we need to set the index if we use keys other than the first key.

Notatka: You must replace the `0x3456789012` with the key configured for use on the access point.

You are encouraged to read `ifconfig(8)` manual page for further information.

The `wpa_supplicant` facility also can be used to configure your wireless interface with WEP. The example above can be set up by adding the following lines to `/etc/wpa_supplicant.conf`:

```
network={
    ssid="my_net"
    key_mgmt=NONE
    wep_key3=3456789012
    wep_tx_keyidx=3
}
```

Then:

```
# wpa_supplicant -i ath0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:49:41:76 (SSID='dlinkap' freq=2437 MHz)
Associated with 00:13:46:49:41:76
```

27.3.4. Ad-hoc Mode

IBSS mode, also called ad-hoc mode, is designed for point to point connections. For example, to establish an ad-hoc network between the machine A and the machine B we will just need to choose two IP addresses and a SSID.

On the box A:

```
# ifconfig ath0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mediaopt adhoc
# ifconfig ath0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.0.1 netmask 0xffffffff broadcast 192.168.0.255
    inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid 0x4
    ether 00:11:95:c3:0d:ac
    media: IEEE 802.11 Wireless Ethernet autoselect <adhoc> (autoselect <adhoc>)
    status: associated
    ssid freebsdap channel 2 bssid 02:11:95:c3:0d:ac
    authmode OPEN privacy OFF txpowmax 36 protmode CTS bintval 100
```

The `adhoc` parameter indicates the interface is running in the IBSS mode.

On B, we should be able to detect A:

```
# ifconfig ath0 up scan
SSID          BSSID          CHAN RATE  S:N   INT CAPS
freebsdap     02:11:95:c3:0d:ac    2   54M 19:0   100 IS
```

The `1` in the output confirms the machine A is in ad-hoc mode. We just have to configure B with a different IP address:

```
# ifconfig ath0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap mediaopt adhoc
# ifconfig ath0
ath0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet6 fe80::211:95ff:fed5:4362%ath0 prefixlen 64 scopeid 0x1
    inet 192.168.0.2 netmask 0xffffffff00 broadcast 192.168.0.255
    ether 00:11:95:d5:43:62
    media: IEEE 802.11 Wireless Ethernet autoselect <adhoc> (autoselect <adhoc>)
    status: associated
    ssid freebsdap channel 2 bssid 02:11:95:c3:0d:ac
    authmode OPEN privacy OFF txpowmax 36 protmode CTS bintval 100
```

Both A and B are now ready to exchange informations.

27.3.5. Troubleshooting

If you are having trouble with wireless networking, there are a number of steps you can take to help troubleshoot the problem.

- If you do not see the access point listed when scanning be sure you have not configured your wireless device to a limited set of channels.
- If you cannot associate to an access point verify the configuration of your station matches the one of the access point. This includes the authentication scheme and any security protocols. Simplify your configuration as much as possible. If you are using a security protocol such as WPA or WEP configure the access point for open authentication and no security to see if you can get traffic to pass.

- Once you can associate to the access point diagnose any security configuration using simple tools like `ping(8)`.

The `wpa_supplicant` has much debugging support; try running it manually with the `-dd` option and look at the system logs.

- There are also many lower-level debugging tools. You can enable debugging messages in the 802.11 protocol support layer using the `wldebug` program found in `/usr/src/tools/tools/net80211`. For example:

```
# wldebug -i ath0 +scan+auth+debug+assoc
net.wlan.0.debug: 0 => 0xc80000<assoc,auth,scan>
```

can be used to enable console messages related to scanning for access points and doing the 802.11 protocol handshakes required to arrange communication.

There are also many useful statistics maintained by the 802.11 layer; the `wlanstats` tool will dump these informations. These statistics should identify all errors identified by the 802.11 layer. Beware however that some errors are identified in the device drivers that lie below the 802.11 layer so they may not show up. To diagnose device-specific problems you need to refer to the drivers' documentation.

If the above information does not help to clarify the problem, please submit a problem report and include output from the above tools.

27.4. Bluetooth

Written by Pav Lucistnik.

27.4.1. Introduction

Bluetooth is a wireless technology for creating personal networks operating in the 2.4 GHz unlicensed band, with a range of 10 meters. Networks are usually formed ad-hoc from portable devices such as cellular phones, handhelds and laptops. Unlike the other popular wireless technology, Wi-Fi, Bluetooth offers higher level service profiles, e.g. FTP-like file servers, file pushing, voice transport, serial line emulation, and more.

The Bluetooth stack in FreeBSD is implemented using the Netgraph framework (see `netgraph(4)`). A broad variety of Bluetooth USB dongles is supported by the `ng_ubt(4)` driver. The Broadcom BCM2033 chip based Bluetooth devices are supported via the `ubtbcmfw(4)` and `ng_ubt(4)` drivers. The 3Com Bluetooth PC Card 3CRWB60-A is supported by the `ng_bt3c(4)` driver. Serial and UART based Bluetooth devices are supported via `sio(4)`, `ng_h4(4)` and `hcseriald(8)`. This section describes the use of the USB Bluetooth dongle.

27.4.2. Plugging in the Device

By default Bluetooth device drivers are available as kernel modules. Before attaching a device, you will need to load the driver into the kernel:

```
# kldload ng_ubt
```

If the Bluetooth device is present in the system during system startup, load the module from `/boot/loader.conf`:

```
ng_ubt_load="YES"
```

Plug in your USB dongle. The output similar to the following will appear on the console (or in syslog):

```
ubt0: vendor 0x0a12 product 0x0001, rev 1.10/5.25, addr 2
ubt0: Interface 0 endpoints: interrupt=0x81, bulk-in=0x82, bulk-out=0x2
ubt0: Interface 1 (alt.config 5) endpoints: isoc-in=0x83, isoc-out=0x3,
      wMaxPacketSize=49, nframes=6, buffer size=294
```

Notatka: The Bluetooth stack has to be started manually on FreeBSD 6.0, and on FreeBSD 5.X before 5.5. It is done automatically from `devd(8)` on FreeBSD 5.5, 6.1 and newer.

Copy `/usr/share/examples/netgraph/bluetooth/rc.bluetooth` into some convenient place, like `/etc/rc.bluetooth`. This script is used to start and stop the Bluetooth stack. It is a good idea to stop the stack before unplugging the device, but it is not (usually) fatal. When starting the stack, you will receive output similar to the following:

```
# /etc/rc.bluetooth start ubt0
BD_ADDR: 00:02:72:00:d4:1a
Features: 0xff 0xff 0xf 00 00 00 00 00
<3-Slot> <5-Slot> <Encryption> <Slot offset>
<Timing accuracy> <Switch> <Hold mode> <Sniff mode>
<Park mode> <RSSI> <Channel quality> <SCO link>
<HV2 packets> <HV3 packets> <u-law log> <A-law log> <CVSD>
<Paging scheme> <Power control> <Transparent SCO data>
Max. ACL packet size: 192 bytes
Number of ACL packets: 8
```

```
Max. SCO packet size: 64 bytes
Number of SCO packets: 8
```

27.4.3. Host Controller Interface (HCI)

Host Controller Interface (HCI) provides a command interface to the baseband controller and link manager, and access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband capabilities. HCI layer on the Host exchanges data and commands with the HCI firmware on the Bluetooth hardware. The Host Controller Transport Layer (i.e. physical bus) driver provides both HCI layers with the ability to exchange information with each other.

A single Netgraph node of type *hci* is created for a single Bluetooth device. The HCI node is normally connected to the Bluetooth device driver node (downstream) and the L2CAP node (upstream). All HCI operations must be performed on the HCI node and not on the device driver node. Default name for the HCI node is „devicehci”. For more details refer to the `ng_hci(4)` manual page.

One of the most common tasks is discovery of Bluetooth devices in RF proximity. This operation is called *inquiry*. Inquiry and other HCI related operations are done with the `hccontrol(8)` utility. The example below shows how to find out which Bluetooth devices are in range. You should receive the list of devices in a few seconds. Note that a remote device will only answer the inquiry if it put into *discoverable* mode.

```
% hccontrol -n ubt0hci inquiry
Inquiry result, num_responses=1
Inquiry result #0
    BD_ADDR: 00:80:37:29:19:a4
    Page Scan Rep. Mode: 0x1
    Page Scan Period Mode: 00
    Page Scan Mode: 00
    Class: 52:02:04
    Clock offset: 0x78ef
Inquiry complete. Status: No error [00]
```

BD_ADDR is unique address of a Bluetooth device, similar to MAC addresses of a network card. This address is needed for further communication with a device. It is possible to assign human readable name to a BD_ADDR. The `/etc/bluetooth/hosts` file contains information regarding the known Bluetooth hosts. The following example shows how to obtain human readable name that was assigned to the remote device:

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav's T39
```

If you perform an inquiry on a remote Bluetooth device, it will find your computer as „your.host.name (ubt0)”. The name assigned to the local device can be changed at any time.

The Bluetooth system provides a point-to-point connection (only two Bluetooth units involved), or a point-to-multipoint connection. In the point-to-multipoint connection the connection is shared among several Bluetooth devices. The following example shows how to obtain the list of active baseband connections for the local device:

```
% hccontrol -n ubt0hci read_connection_list
```

Remote	BD_ADDR	Handle	Type	Mode	Role	Encrypt	Pending	Queue	State
00:80:37:29:19:a4		41	ACL	0	MAST	NONE	0	0	OPEN

A *connection handle* is useful when termination of the baseband connection is required. Note, that it is normally not required to do it by hand. The stack will automatically terminate inactive baseband connections.

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
Reason: Connection terminated by local host [0x16]
```

Refer to `hccontrol help` for a complete listing of available HCI commands. Most of the HCI commands do not require superuser privileges.

27.4.4. Logical Link Control and Adaptation Protocol (L2CAP)

Logical Link Control and Adaptation Protocol (L2CAP) provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability and segmentation and reassembly operation. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

L2CAP is based around the concept of *channels*. Channel is a logical connection on top of baseband connection. Each channel is bound to a single protocol in a many-to-one fashion. Multiple channels can be bound to the same protocol, but a channel cannot be bound to multiple protocols. Each L2CAP packet received on a channel is directed to the appropriate higher level protocol. Multiple channels can share the same baseband connection.

A single Netgraph node of type *l2cap* is created for a single Bluetooth device. The L2CAP node is normally connected to the Bluetooth HCI node (downstream) and Bluetooth sockets nodes (upstream). Default name for the L2CAP node is „device12cap”. For more details refer to the `ng_l2cap(4)` manual page.

A useful command is `l2ping(8)`, which can be used to ping other devices. Some Bluetooth implementations might not return all of the data sent to them, so 0 bytes in the following example is normal.

```
# l2ping -a 00:80:37:29:19:a4
0 bytes from 0:80:37:29:19:a4 seq_no=0 time=48.633 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=1 time=37.551 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=2 time=28.324 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=3 time=46.150 ms result=0
```

The `l2control(8)` utility is used to perform various operations on L2CAP nodes. This example shows how to obtain the list of logical connections (channels) and the list of baseband connections for the local device:

```
% l2control -a 00:02:72:00:d4:1a read_channel_list
L2CAP channels:
Remote BD_ADDR      SCID/ DCID   PSM  IMTU/ OMTU State
00:07:e0:00:0b:ca    66/   64     3   132/  672 OPEN
% l2control -a 00:02:72:00:d4:1a read_connection_list
L2CAP connections:
Remote BD_ADDR      Handle Flags Pending State
00:07:e0:00:0b:ca    41  0           0 OPEN
```

Another diagnostic tool is `btsockstat(1)`. It does a job similar to as `netstat(1)` does, but for Bluetooth network-related data structures. The example below shows the same logical connection as `l2control(8)` above.

```
% btsockstat
Active L2CAP sockets
PCB      Recv-Q Send-Q Local address/PSM      Foreign address  CID   State
c2afe900      0      0 00:02:72:00:d4:1a/3    00:07:e0:00:0b:ca 66    OPEN
Active RFCOMM sessions
L2PCB     PCB      Flag MTU   Out-Q DLCs State
c2afe900 c2b53380 1     127     0     Yes  OPEN
Active RFCOMM sockets
PCB      Recv-Q Send-Q Local address      Foreign address  Chan DLCI State
c2e8bc80      0     250 00:02:72:00:d4:1a 00:07:e0:00:0b:ca 3      6    OPEN
```

27.4.5. RFCOMM Protocol

The RFCOMM protocol provides emulation of serial ports over the L2CAP protocol. The protocol is based on the ETSI standard TS 07.10. RFCOMM is a simple transport protocol, with additional provisions for emulating the 9 circuits of RS-232 (EIA/TIA-232-E) serial ports. The RFCOMM protocol supports up to 60 simultaneous connections (RFCOMM channels) between two Bluetooth devices.

For the purposes of RFCOMM, a complete communication path involves two applications running on different devices (the communication endpoints) with a communication segment between them. RFCOMM is intended to cover applications that make use of the serial ports of the devices in which they reside. The communication segment is a Bluetooth link from one device to another (direct connect).

RFCOMM is only concerned with the connection between the devices in the direct connect case, or between the device and a modem in the network case. RFCOMM can support other configurations, such as modules that communicate via Bluetooth wireless technology on one side and provide a wired interface on the other side.

In FreeBSD the RFCOMM protocol is implemented at the Bluetooth sockets layer.

27.4.6. Pairing of Devices

By default, Bluetooth communication is not authenticated, and any device can talk to any other device. A Bluetooth device (for example, cellular phone) may choose to require authentication to provide a particular service (for example, Dial-Up service). Bluetooth authentication is normally done with *PIN codes*. A PIN code is an ASCII string up to 16 characters in length. User is required to enter the same PIN code on both devices. Once user has entered the PIN code, both devices will generate a *link key*. After that the link key can be stored either in the devices themselves or in a persistent storage. Next time both devices will use previously generated link key. The described above procedure is called *pairing*. Note that if the link key is lost by any device then pairing must be repeated.

The hcsecd(8) daemon is responsible for handling of all Bluetooth authentication requests. The default configuration file is /etc/bluetooth/hcsecd.conf. An example section for a cellular phone with the PIN code arbitrarily set to „1234” is shown below:

```
device {
    bdaddr 00:80:37:29:19:a4;
    name    "Pav's T39";
    key     nokey;
    pin     "1234";
}
```

There is no limitation on PIN codes (except length). Some devices (for example Bluetooth headsets) may have a fixed PIN code built in. The `-d` switch forces the `hcsecd(8)` daemon to stay in the foreground, so it is easy to see what is happening. Set the remote device to receive pairing and initiate the Bluetooth connection to the remote device. The remote device should say that pairing was accepted, and request the PIN code. Enter the same PIN code as you have in `hcsecd.conf`. Now your PC and the remote device are paired. Alternatively, you can initiate pairing on the remote device.

On FreeBSD 5.5, 6.1 and newer, the following line can be added to the `/etc/rc.conf` file to have **hcsecd** started automatically on system start:

```
hcsecd_enable="YES"
```

The following is a sample of the **hcsecd** daemon output:

```
hcsecd[16484]: Got Link_Key_Request event from 'ubt0hci', remote bdaddr 0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39', link key d
hcsecd[16484]: Sending Link_Key_Negative_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
hcsecd[16484]: Got PIN_Code_Request event from 'ubt0hci', remote bdaddr 0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39', PIN code e
hcsecd[16484]: Sending PIN_Code_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
```

27.4.7. Service Discovery Protocol (SDP)

The Service Discovery Protocol (SDP) provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to utilize the service.

SDP involves communication between a SDP server and a SDP client. The server maintains a list of service records that describe the characteristics of services associated with the server. Each service record contains information about a single service. A client may retrieve information from a service record maintained by the SDP server by issuing a SDP request. If the client, or an application associated with the client, decides to use a service, it must open a separate connection to the service provider in order to utilize the service. SDP provides a mechanism for discovering services and their attributes, but it does not provide a mechanism for utilizing those services.

Normally, a SDP client searches for services based on some desired characteristics of the services. However, there are times when it is desirable to discover which types of services are described by an SDP server's service records without any a priori information about the services. This process of looking for any offered services is called *browsing*.

The Bluetooth SDP server `sdpd(8)` and command line client `sdpcontrol(8)` are included in the standard FreeBSD installation. The following example shows how to perform a SDP browse query.

```
% sdpcontrol -a 00:01:03:fc:6e:ec browse
Record Handle: 00000000
Service Class ID List:
    Service Discovery Server (0x1000)
Protocol Descriptor List:
    L2CAP (0x0100)
        Protocol specific parameter #1: u/int/uuid16 1
        Protocol specific parameter #2: u/int/uuid16 1

Record Handle: 0x00000001
```

```

Service Class ID List:
    Browse Group Descriptor (0x1001)

Record Handle: 0x00000002
Service Class ID List:
    LAN Access Using PPP (0x1102)
Protocol Descriptor List:
    L2CAP (0x0100)
    RFCOMM (0x0003)
        Protocol specific parameter #1: u/int8/bool 1
Bluetooth Profile Descriptor List:
    LAN Access Using PPP (0x1102) ver. 1.0

```

... and so on. Note that each service has a list of attributes (RFCOMM channel for example). Depending on the service you might need to make a note of some of the attributes. Some Bluetooth implementations do not support service browsing and may return an empty list. In this case it is possible to search for the specific service. The example below shows how to search for the OBEX Object Push (OPUSH) service:

```
% sdpcontrol -a 00:01:03:fc:6e:ec search OPUSH
```

Offering services on FreeBSD to Bluetooth clients is done with the `sdpd(8)` server. On FreeBSD 5.5, 6.1 and newer, the following line can be added to the `/etc/rc.conf` file:

```
sdpd_enable="YES"
```

Then the **sdpd** daemon can be started with:

```
# /etc/rc.d/sdpd start
```

On FreeBSD 6.0, and on FreeBSD 5.X before 5.5, **sdpd** is not integrated into the system startup scripts. It has to be started manually with:

```
# sdpd
```

The local server application that wants to provide Bluetooth service to the remote clients will register service with the local SDP daemon. The example of such application is `rfcomm_pppd(8)`. Once started it will register Bluetooth LAN service with the local SDP daemon.

The list of services registered with the local SDP server can be obtained by issuing SDP browse query via local control channel:

```
# sdpcontrol -l browse
```

27.4.8. Dial-Up Networking (DUN) and Network Access with PPP (LAN) Profiles

The Dial-Up Networking (DUN) profile is mostly used with modems and cellular phones. The scenarios covered by this profile are the following:

- use of a cellular phone or modem by a computer as a wireless modem for connecting to a dial-up Internet access server, or using other dial-up services;
- use of a cellular phone or modem by a computer to receive data calls.

Network Access with PPP (LAN) profile can be used in the following situations:

- LAN access for a single Bluetooth device;
- LAN access for multiple Bluetooth devices;
- PC to PC (using PPP networking over serial cable emulation).

In FreeBSD both profiles are implemented with `ppp(8)` and `rfcomm_pppd(8)` - a wrapper that converts RFCOMM Bluetooth connection into something PPP can operate with. Before any profile can be used, a new PPP label in the `/etc/ppp/ppp.conf` must be created. Consult `rfcomm_pppd(8)` manual page for examples.

In the following example `rfcomm_pppd(8)` will be used to open RFCOMM connection to remote device with BD_ADDR 00:80:37:29:19:a4 on DUN RFCOMM channel. The actual RFCOMM channel number will be obtained from the remote device via SDP. It is possible to specify RFCOMM channel by hand, and in this case `rfcomm_pppd(8)` will not perform SDP query. Use `sdpcontrol(8)` to find out RFCOMM channel on the remote device.

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

In order to provide Network Access with PPP (LAN) service the `sdpd(8)` server must be running. A new entry for LAN clients must be created in the `/etc/ppp/ppp.conf` file. Consult `rfcomm_pppd(8)` manual page for examples. Finally, start RFCOMM PPP server on valid RFCOMM channel number. The RFCOMM PPP server will automatically register Bluetooth LAN service with the local SDP daemon. The example below shows how to start RFCOMM PPP server.

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

27.4.9. OBEX Object Push (OPUSH) Profile

OBEX is a widely used protocol for simple file transfers between mobile devices. Its main use is in infrared communication, where it is used for generic file transfers between notebooks or PDAs, and for sending business cards or calendar entries between cellular phones and other devices with PIM applications.

The OBEX server and client are implemented as a third-party package **obexapp**, which is available as `comms/obexapp` port.

OBEX client is used to push and/or pull objects from the OBEX server. An object can, for example, be a business card or an appointment. The OBEX client can obtain RFCOMM channel number from the remote device via SDP. This can be done by specifying service name instead of RFCOMM channel number. Supported service names are: IrMC, FTRN and OPUSH. It is possible to specify RFCOMM channel as a number. Below is an example of an OBEX session, where device information object is pulled from the cellular phone, and a new object (business card) is pushed into the phone's directory.

```
% obexapp -a 00:80:37:29:19:a4 -C IrMC
obex> get telecom/devinfo.txt devinfo-t39.txt
Success, response: OK, Success (0x20)
obex> put new.vcf
Success, response: OK, Success (0x20)
obex> di
Success, response: OK, Success (0x20)
```

In order to provide OBEX Object Push service, `sdpd(8)` server must be running. A root folder, where all incoming objects will be stored, must be created. The default path to the root folder is `/var/spool/obex`. Finally, start OBEX server on valid RFCOMM channel number. The OBEX server will automatically register OBEX Object Push service with the local SDP daemon. The example below shows how to start OBEX server.

```
# obexapp -s -C 10
```

27.4.10. Serial Port Profile (SPP)

The Serial Port Profile (SPP) allows Bluetooth devices to perform RS232 (or similar) serial cable emulation. The scenario covered by this profile deals with legacy applications using Bluetooth as a cable replacement, through a virtual serial port abstraction.

The `rfcomm_sppd(1)` utility implements the Serial Port profile. A pseudo tty is used as a virtual serial port abstraction. The example below shows how to connect to a remote device Serial Port service. Note that you do not have to specify a RFCOMM channel - `rfcomm_sppd(1)` can obtain it from the remote device via SDP. If you would like to override this, specify a RFCOMM channel on the command line.

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t /dev/ttyp6
rfcomm_sppd[94692]: Starting on /dev/ttyp6...
```

Once connected, the pseudo tty can be used as serial port:

```
# cu -l ttyp6
```

27.4.11. Troubleshooting

27.4.11.1. A remote device cannot connect

Some older Bluetooth devices do not support role switching. By default, when FreeBSD is accepting a new connection, it tries to perform a role switch and become master. Devices, which do not support this will not be able to connect. Note that role switching is performed when a new connection is being established, so it is not possible to ask the remote device if it does support role switching. There is a HCI option to disable role switching on the local side:

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

27.4.11.2. Something is going wrong, can I see what exactly is happening?

Yes, you can. Use the third-party package **hcidump**, which is available as `comms/hcidump` port. The **hcidump** utility is similar to `tcpdump(1)`. It can be used to display the content of the Bluetooth packets on the terminal and to dump the Bluetooth packets to a file.

27.5. Bridging

Written by Steve Peterson.

27.5.1. Introduction

It is sometimes useful to divide one physical network (such as an Ethernet segment) into two separate network segments without having to create IP subnets and use a router to connect the segments together. A device that connects two networks together in this fashion is called a „bridge”. A FreeBSD system with two network interface cards can act as a bridge.

The bridge works by learning the MAC layer addresses (Ethernet addresses) of the devices on each of its network interfaces. It forwards traffic between two networks only when its source and destination are on different networks.

In many respects, a bridge is like an Ethernet switch with very few ports.

27.5.2. Situations Where Bridging Is Appropriate

There are two common situations in which a bridge is used today.

27.5.2.1. High Traffic on a Segment

Situation one is where your physical network segment is overloaded with traffic, but you do not want for whatever reason to subnet the network and interconnect the subnets with a router.

Let us consider an example of a newspaper where the Editorial and Production departments are on the same subnetwork. The Editorial users all use server A for file service, and the Production users are on server B. An Ethernet network is used to connect all users together, and high loads on the network are slowing things down.

If the Editorial users could be segregated on one network segment and the Production users on another, the two network segments could be connected with a bridge. Only the network traffic destined for interfaces on the „other” side of the bridge would be sent to the other network, reducing congestion on each network segment.

27.5.2.2. Filtering/Traffic Shaping Firewall

The second common situation is where firewall functionality is needed without network address translation (NAT).

An example is a small company that is connected via DSL or ISDN to their ISP. They have a 13 globally-accessible IP addresses from their ISP and have 10 PCs on their network. In this situation, using a router-based firewall is difficult because of subnetting issues.

A bridge-based firewall can be configured and dropped into the path just downstream of their DSL/ISDN router without any IP numbering issues.

27.5.3. Configuring a Bridge

27.5.3.1. Network Interface Card Selection

A bridge requires at least two network cards to function. Unfortunately, not all network interface cards support bridging. Read `bridge(4)` for details on the cards that are supported.

Install and test the two network cards before continuing.

27.5.3.2. Kernel Configuration Changes

To enable kernel support for bridging, add the:

```
options BRIDGE
```

statement to your kernel configuration file, and rebuild your kernel.

27.5.3.3. Firewall Support

If you are planning to use the bridge as a firewall, you will need to add the `IPFIREWALL` option as well. Read [Rozdział 26](#) for general information on configuring the bridge as a firewall.

If you need to allow non-IP packets (such as ARP) to flow through the bridge, there is a firewall option that must be set. This option is `IPFIREWALL_DEFAULT_TO_ACCEPT`. Note that this changes the default rule for the firewall to accept any packet. Make sure you know how this changes the meaning of your ruleset before you set it.

27.5.3.4. Traffic Shaping Support

If you want to use the bridge as a traffic shaper, you will need to add the `DUMMYNET` option to your kernel configuration. Read `dummynet(4)` for further information.

27.5.4. Enabling the Bridge

Add the line:

```
net.link.ether.bridge.enable=1
```

to `/etc/sysctl.conf` to enable the bridge at runtime, and the line:

```
net.link.ether.bridge.config=if1,if2
```

to enable bridging on the specified interfaces (replace `if1` and `if2` with the names of your two network interfaces).

If you want the bridged packets to be filtered by `ipfw(8)`, you should add:

```
net.link.ether.bridge.ipfw=1
```

as well.

For versions prior to FreeBSD 5.2-RELEASE, use instead the following lines:

```
net.link.ether.bridge=1
net.link.ether.bridge_cfg=if1,if2
net.link.ether.bridge_ipfw=1
```

27.5.5. Other Information

If you want to be able to `ssh(1)` into the bridge from the network, it is correct to assign one of the network cards an IP address. The consensus is that assigning both cards an address is a bad idea.

If you have multiple bridges on your network, there cannot be more than one path between any two workstations. Technically, this means that there is no support for spanning tree link management.

A bridge can add latency to your `ping(8)` times, especially for traffic from one segment to another.

27.6. Diskless Operation

Updated by Jean-François Dockès. Reorganized and enhanced by Alex Dupre.

A FreeBSD machine can boot over the network and operate without a local disk, using file systems mounted from an NFS server. No system modification is necessary, beyond standard configuration files. Such a system is relatively easy to set up because all the necessary elements are readily available:

- There are at least two possible methods to load the kernel over the network:
 - PXE: The Intel Preboot eXecution Environment system is a form of smart boot ROM built into some networking cards or motherboards. See `pxeboot(8)` for more details.
 - The **Etherboot** port (`net/etherboot`) produces ROM-able code to boot kernels over the network. The code can be either burnt into a boot PROM on a network card, or loaded from a local floppy (or hard) disk drive, or from a running MS-DOS system. Many network cards are supported.
- A sample script (`/usr/share/examples/diskless/clone_root`) eases the creation and maintenance of the workstation's root file system on the server. The script will probably require a little customization but it will get you started very quickly.
- Standard system startup files exist in `/etc` to detect and support a diskless system startup.
- Swapping, if needed, can be done either to an NFS file or to a local disk.

There are many ways to set up diskless workstations. Many elements are involved, and most can be customized to suit local taste. The following will describe variations on the setup of a complete system, emphasizing simplicity and compatibility with the standard FreeBSD startup scripts. The system described has the following characteristics:

- The diskless workstations use a shared read-only `/` file system, and a shared read-only `/usr`.

The root file system is a copy of a standard FreeBSD root (typically the server's), with some configuration files overridden by ones specific to diskless operation or, possibly, to the workstation they belong to.

The parts of the root which have to be writable are overlaid with `md(4)` file systems. Any changes will be lost when the system reboots.

- The kernel is transferred and loaded either with **Etherboot** or PXE as some situations may mandate the use of either method.

Uwaga!As described, this system is insecure. It should live in a protected area of a network, and be untrusted by other hosts.

All the information in this section has been tested using FreeBSD 5.2.1-RELEASE.

27.6.1. Background Information

Setting up diskless workstations is both relatively straightforward and prone to errors. These are sometimes difficult to diagnose for a number of reasons. For example:

- Compile time options may determine different behaviors at runtime.
- Error messages are often cryptic or totally absent.

In this context, having some knowledge of the background mechanisms involved is very useful to solve the problems that may arise.

Several operations need to be performed for a successful bootstrap:

- The machine needs to obtain initial parameters such as its IP address, executable filename, server name, root path. This is done using the DHCP or BOOTP protocols. DHCP is a compatible extension of BOOTP, and uses the same port numbers and basic packet format.

It is possible to configure a system to use only BOOTP. The bootpd(8) server program is included in the base FreeBSD system.

However, DHCP has a number of advantages over BOOTP (nicer configuration files, possibility of using PXE, plus many others not directly related to diskless operation), and we will describe mainly a DHCP configuration, with equivalent examples using bootpd(8) when possible. The sample configuration will use the **ISC DHCP** software package (release 3.0.1.r12 was installed on the test server).

- The machine needs to transfer one or several programs to local memory. Either TFTP or NFS are used. The choice between TFTP and NFS is a compile time option in several places. A common source of error is to specify filenames for the wrong protocol: TFTP typically transfers all files from a single directory on the server, and would expect filenames relative to this directory. NFS needs absolute file paths.
- The possible intermediate bootstrap programs and the kernel need to be initialized and executed. There are several important variations in this area:
 - PXE will load pxeboot(8), which is a modified version of the FreeBSD third stage loader. The loader(8) will obtain most parameters necessary to system startup, and leave them in the kernel environment before transferring control. It is possible to use a **GENERIC** kernel in this case.
 - **Etherboot**, will directly load the kernel, with less preparation. You will need to build a kernel with specific options.

PXE and **Etherboot** work equally well; however, because kernels normally let the loader(8) do more work for them, PXE is the preferred method.

If your BIOS and network cards support PXE, you should probably use it.

- Finally, the machine needs to access its file systems. NFS is used in all cases.

See also `diskless(8)` manual page.

27.6.2. Setup Instructions

27.6.2.1. Configuration Using ISC DHCP

The **ISC DHCP** server can answer both BOOTP and DHCP requests.

ISC DHCP 3.0 is not part of the base system. You will first need to install the `net/isc-dhcp3-server` port or the corresponding package.

Once **ISC DHCP** is installed, it needs a configuration file to run, (normally named `/usr/local/etc/dhcpd.conf`). Here follows a commented example, where host `margaux` uses **Etherboot** and host `corbieres` uses PXE:

```
default-lease-time 600;
max-lease-time 7200;
authoritative;

option domain-name "example.com";
option domain-name-servers 192.168.4.1;
option routers 192.168.4.1;

subnet 192.168.4.0 netmask 255.255.255.0 {
    use-host-decl-names on; ❶
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.4.255;

    host margaux {
        hardware ethernet 01:23:45:67:89:ab;
        fixed-address margaux.example.com;
        next-server 192.168.4.4; ❷
        filename "/data/misc/kernel.diskless"; ❸
        option root-path "192.168.4.4:/data/misc/diskless"; ❹
    }
    host corbieres {
        hardware ethernet 00:02:b3:27:62:df;
        fixed-address corbieres.example.com;
        next-server 192.168.4.4;
        filename "pxeboot";
        option root-path "192.168.4.4:/data/misc/diskless";
    }
}
```

- ❶ This option tells **dhcpd** to send the value in the host declarations as the hostname for the diskless host. An alternate way would be to add an option `host-name margaux` inside the host declarations.
- ❷ The `next-server` directive designates the TFTP or NFS server to use for loading loader or kernel file (the default is to use the same host as the DHCP server).

- ③ The `filename` directive defines the file that **Etherboot** or PXE will load for the next execution step. It must be specified according to the transfer method used. **Etherboot** can be compiled to use NFS or TFTP. The FreeBSD port configures NFS by default. PXE uses TFTP, which is why a relative filename is used here (this may depend on the TFTP server configuration, but would be fairly typical). Also, PXE loads `pxeboot`, not the kernel. There are other interesting possibilities, like loading `pxeboot` from a FreeBSD CD-ROM `/boot` directory (as `pxeboot(8)` can load a `GENERIC` kernel, this makes it possible to use PXE to boot from a remote CD-ROM).
- ④ The `root-path` option defines the path to the root file system, in usual NFS notation. When using PXE, it is possible to leave off the host's IP as long as you do not enable the kernel option `BOOTP`. The NFS server will then be the same as the TFTP one.

27.6.2.2. Configuration Using BOOTP

Here follows an equivalent **bootpd** configuration (reduced to one client). This would be found in `/etc/bootptab`.

Please note that **Etherboot** must be compiled with the non-default option `NO_DHCP_SUPPORT` in order to use BOOTP, and that PXE *needs* DHCP. The only obvious advantage of **bootpd** is that it exists in the base system.

```
.def100:\
:hn:ht=1:sa=192.168.4.4:vm=rfc1048:\
:sm=255.255.255.0:\
:ds=192.168.4.1:\
:gw=192.168.4.1:\
:hd="/tftpboot":\
:bf="/kernel.diskless":\
:rp="192.168.4.4:/data/misc/diskless":
```

```
margaux:ha=0123456789ab:tc=.def100
```

27.6.2.3. Preparing a Boot Program with Etherboot

Etherboot's Web site (<http://etherboot.sourceforge.net>) contains extensive documentation (<http://etherboot.sourceforge.net/doc/html/userman/t1.html>) mainly intended for Linux systems, but nonetheless containing useful information. The following will just outline how you would use **Etherboot** on a FreeBSD system.

You must first install the `net/etherboot` package or port.

You can change the **Etherboot** configuration (i.e. to use TFTP instead of NFS) by editing the `Config` file in the **Etherboot** source directory.

For our setup, we shall use a boot floppy. For other methods (PROM, or MS-DOS program), please refer to the **Etherboot** documentation.

To make a boot floppy, insert a floppy in the drive on the machine where you installed **Etherboot**, then change your current directory to the `src` directory in the **Etherboot** tree and type:

```
# gmake bin32/devicetype.fd0
```

`devicetype` depends on the type of the Ethernet card in the diskless workstation. Refer to the `NIC` file in the same directory to determine the right `devicetype`.

27.6.2.4. Booting with PXE

By default, the pxeboot(8) loader loads the kernel via NFS. It can be compiled to use TFTP instead by specifying the `LOADER_TFTP_SUPPORT` option in `/etc/make.conf`. See the comments in `/usr/share/examples/etc/make.conf` for instructions.

There are two other undocumented `make.conf` options which may be useful for setting up a serial console diskless machine: `BOOT_PXELDR_PROBE_KEYBOARD`, and `BOOT_PXELDR_ALWAYS_SERIAL`.

To use PXE when the machine starts, you will usually need to select the `Boot from network` option in your BIOS setup, or type a function key during the PC initialization.

27.6.2.5. Configuring the TFTP and NFS Servers

If you are using PXE or **Etherboot** configured to use TFTP, you need to enable **tftpd** on the file server:

1. Create a directory from which **tftpd** will serve the files, e.g. `/tftpboot`.

2. Add this line to your `/etc/inetd.conf`:

```
tftp      dgram    udp      wait     root     /usr/libexec/tftpd      tftpd -l -s /tftpboot
```

Notatka: It appears that at least some PXE versions want the TCP version of TFTP. In this case, add a second line, replacing `dgram udp` with `stream tcp`.

3. Tell **inetd** to reread its configuration file. The `inetd_enable="YES"` must be in the `/etc/rc.conf` file for this command to execute correctly:

```
# /etc/rc.d/inetd restart
```

You can place the `tftpboot` directory anywhere on the server. Make sure that the location is set in both `inetd.conf` and `dhcpcd.conf`.

In all cases, you also need to enable NFS and export the appropriate file system on the NFS server.

1. Add this to `/etc/rc.conf`:

```
nfs_server_enable="YES"
```

2. Export the file system where the diskless root directory is located by adding the following to `/etc/exports` (adjust the volume mount point and replace *margaux corbieres* with the names of the diskless workstations):

```
/data/misc -alldirs -ro margaux corbieres
```

3. Tell **mountd** to reread its configuration file. If you actually needed to enable NFS in `/etc/rc.conf` at the first step, you probably want to reboot instead.

```
# /etc/rc.d/mountd restart
```

27.6.2.6. Building a Diskless Kernel

If using **Etherboot**, you need to create a kernel configuration file for the diskless client with the following options (in addition to the usual ones):

```
options      BOOTP          # Use BOOTP to obtain IP address/hostname
options      BOOTP_NFSROOT  # NFS mount root file system using BOOTP info
```

You may also want to use `BOOTP_NFSV3`, `BOOT_COMPAT` and `BOOTP_WIRED_TO` (refer to NOTES).

These option names are historical and slightly misleading as they actually enable indifferent use of DHCP and BOOTP inside the kernel (it is also possible to force strict BOOTP or DHCP use).

Build the kernel (see Rozdział 8), and copy it to the place specified in `dhcpcd.conf`.

Notatka: When using PXE, building a kernel with the above options is not strictly necessary (though suggested). Enabling them will cause more DHCP requests to be issued during kernel startup, with a small risk of inconsistency between the new values and those retrieved by `pxeboot(8)` in some special cases. The advantage of using them is that the host name will be set as a side effect. Otherwise you will need to set the host name by another method, for example in a client-specific `rc.conf` file.

Notatka: In order to be loadable with **Etherboot**, a kernel needs to have the device hints compiled in. You would typically set the following option in the configuration file (see the NOTES configuration comments file):

```
hints          "GENERIC.hints"
```

27.6.2.7. Preparing the Root Filesystem

You need to create a root file system for the diskless workstations, in the location listed as `root-path` in `dhcpcd.conf`.

27.6.2.7.1. Using `make world` to populate root

This method is quick and will install a complete virgin system (not only the root file system) into `DESTDIR`. All you have to do is simply execute the following script:

```
#!/bin/sh
export DESTDIR=/data/misc/diskless
mkdir -p ${DESTDIR}
cd /usr/src; make buildworld && make buildkernel
cd /usr/src/etc; make distribution
```

Once done, you may need to customize your `/etc/rc.conf` and `/etc/fstab` placed into `DESTDIR` according to your needs.

27.6.2.8. Configuring Swap

If needed, a swap file located on the server can be accessed via NFS.

27.6.2.8.1. NFS Swap

The kernel does not support enabling NFS swap at boot time. Swap must be enabled by the startup scripts, by mounting a writable file system and creating and enabling a swap file. To create a swap file of appropriate size, you can do like this:

```
# dd if=/dev/zero of=/path/to/swapfile bs=1k count=1 oseek=100000
```

To enable it you have to add the following line to your `rc.conf`:

```
swapfile=/path/to/swapfile
```

27.6.2.9. Miscellaneous Issues

27.6.2.9.1. Running with a Read-only `/usr`

If the diskless workstation is configured to run X, you will have to adjust the **XDM** configuration file, which puts the error log on `/usr` by default.

27.6.2.9.2. Using a Non-FreeBSD Server

When the server for the root file system is not running FreeBSD, you will have to create the root file system on a FreeBSD machine, then copy it to its destination, using `tar` or `cpio`.

In this situation, there are sometimes problems with the special files in `/dev`, due to differing major/minor integer sizes. A solution to this problem is to export a directory from the non-FreeBSD server, mount this directory onto a FreeBSD machine, and use `devfs(5)` to allocate device nodes transparently for the user.

27.7. ISDN

A good resource for information on ISDN technology and hardware is Dan Kegel's ISDN Page (<http://www.alumni.caltech.edu/~dank/isdn/>).

A quick simple road map to ISDN follows:

- If you live in Europe you might want to investigate the ISDN card section.
- If you are planning to use ISDN primarily to connect to the Internet with an Internet Provider on a dial-up non-dedicated basis, you might look into Terminal Adapters. This will give you the most flexibility, with the fewest problems, if you change providers.
- If you are connecting two LANs together, or connecting to the Internet with a dedicated ISDN connection, you might consider the stand alone router/bridge option.

Cost is a significant factor in determining what solution you will choose. The following options are listed from least expensive to most expensive.

27.7.1. ISDN Cards

Contributed by Hellmuth Michaelis.

FreeBSD's ISDN implementation supports only the DSS1/Q.931 (or Euro-ISDN) standard using passive cards. Some active cards are supported where the firmware also supports other signaling protocols; this also includes the first supported Primary Rate (PRI) ISDN card.

The **isdn4bsd** software allows you to connect to other ISDN routers using either IP over raw HDLC or by using synchronous PPP: either by using kernel PPP with `isppp`, a modified `sppp(4)` driver, or by using userland `ppp(8)`. By using userland `ppp(8)`, channel bonding of two or more ISDN B-channels is possible. A telephone answering machine application is also available as well as many utilities such as a software 300 Baud modem.

Some growing number of PC ISDN cards are supported under FreeBSD and the reports show that it is successfully used all over Europe and in many other parts of the world.

The passive ISDN cards supported are mostly the ones with the Infineon (formerly Siemens) ISAC/HSCX/IPAC ISDN chipsets, but also ISDN cards with chips from Cologne Chip (ISA bus only), PCI cards with Winbond W6692 chips, some cards with the Tiger300/320/ISAC chipset combinations and some vendor specific chipset based cards such as the AVM Fritz!Card PCI V.1.0 and the AVM Fritz!Card PnP.

Currently the active supported ISDN cards are the AVM B1 (ISA and PCI) BRI cards and the AVM T1 PCI PRI cards.

For documentation on **isdn4bsd**, have a look at `/usr/share/examples/isdn/` directory on your FreeBSD system or at the homepage of **isdn4bsd** (<http://www.freebsd-support.de/i4b/>) which also has pointers to hints, erratas and much more documentation such as the **isdn4bsd** handbook (<http://people.FreeBSD.org/~hm/>).

In case you are interested in adding support for a different ISDN protocol, a currently unsupported ISDN PC card or otherwise enhancing **isdn4bsd**, please get in touch with Hellmuth Michaelis <hm@FreeBSD.org>.

For questions regarding the installation, configuration and troubleshooting **isdn4bsd**, a `freebsd-isdn` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn>) mailing list is available.

27.7.2. ISDN Terminal Adapters

Terminal adapters (TA), are to ISDN what modems are to regular phone lines.

Most TA's use the standard Hayes modem AT command set, and can be used as a drop in replacement for a modem.

A TA will operate basically the same as a modem except connection and throughput speeds will be much faster than your old modem. You will need to configure PPP exactly the same as for a modem setup. Make sure you set your serial speed as high as possible.

The main advantage of using a TA to connect to an Internet Provider is that you can do Dynamic PPP. As IP address space becomes more and more scarce, most providers are not willing to provide you with a static IP anymore. Most stand-alone routers are not able to accommodate dynamic IP allocation.

TA's completely rely on the PPP daemon that you are running for their features and stability of connection. This allows you to upgrade easily from using a modem to ISDN on a FreeBSD machine, if you already have PPP set up. However, at the same time any problems you experienced with the PPP program and are going to persist.

If you want maximum stability, use the kernel PPP option, not the userland PPP.

The following TA's are known to work with FreeBSD:

- Motorola BitSurfer and Bitsurfer Pro
- Adtran

Most other TA's will probably work as well, TA vendors try to make sure their product can accept most of the standard modem AT command set.

The real problem with external TA's is that, like modems, you need a good serial card in your computer.

You should read the FreeBSD Serial Hardware

(http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/serial-uart/index.html) tutorial for a detailed understanding of serial devices, and the differences between asynchronous and synchronous serial ports.

A TA running off a standard PC serial port (asynchronous) limits you to 115.2 Kbs, even though you have a 128 Kbs connection. To fully utilize the 128 Kbs that ISDN is capable of, you must move the TA to a synchronous serial card.

Do not be fooled into buying an internal TA and thinking you have avoided the synchronous/asynchronous issue. Internal TA's simply have a standard PC serial port chip built into them. All this will do is save you having to buy another serial cable and find another empty electrical socket.

A synchronous card with a TA is at least as fast as a stand-alone router, and with a simple 386 FreeBSD box driving it, probably more flexible.

The choice of synchronous card/TA v.s. stand-alone router is largely a religious issue. There has been some discussion of this in the mailing lists. We suggest you search the archives (<http://www.FreeBSD.org/search/index.html>) for the complete discussion.

27.7.3. Stand-alone ISDN Bridges/Routers

ISDN bridges or routers are not at all specific to FreeBSD or any other operating system. For a more complete description of routing and bridging technology, please refer to a networking reference book.

In the context of this section, the terms router and bridge will be used interchangeably.

As the cost of low end ISDN routers/bridges comes down, it will likely become a more and more popular choice. An ISDN router is a small box that plugs directly into your local Ethernet network, and manages its own connection to the other bridge/router. It has built in software to communicate via PPP and other popular protocols.

A router will allow you much faster throughput than a standard TA, since it will be using a full synchronous ISDN connection.

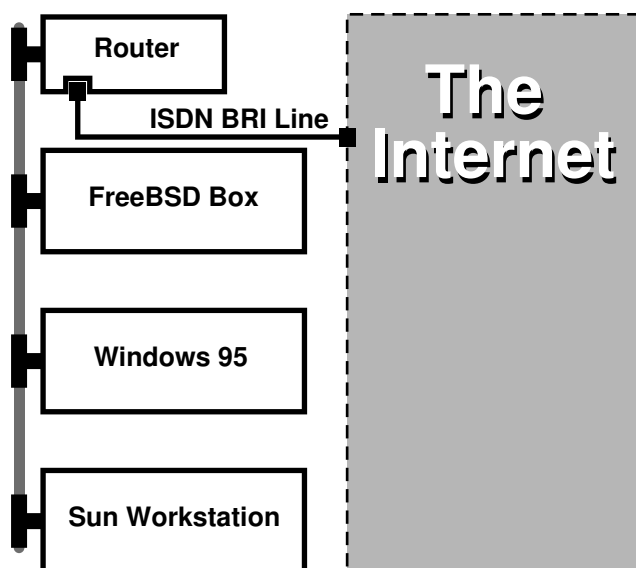
The main problem with ISDN routers and bridges is that interoperability between manufacturers can still be a problem. If you are planning to connect to an Internet provider, you should discuss your needs with them.

If you are planning to connect two LAN segments together, such as your home LAN to the office LAN, this is the simplest lowest maintenance solution. Since you are buying the equipment for both sides of the connection you can be assured that the link will work.

For example to connect a home computer or branch office network to a head office network the following setup could be used:

Przykład 27-1. Branch Office or Home Network

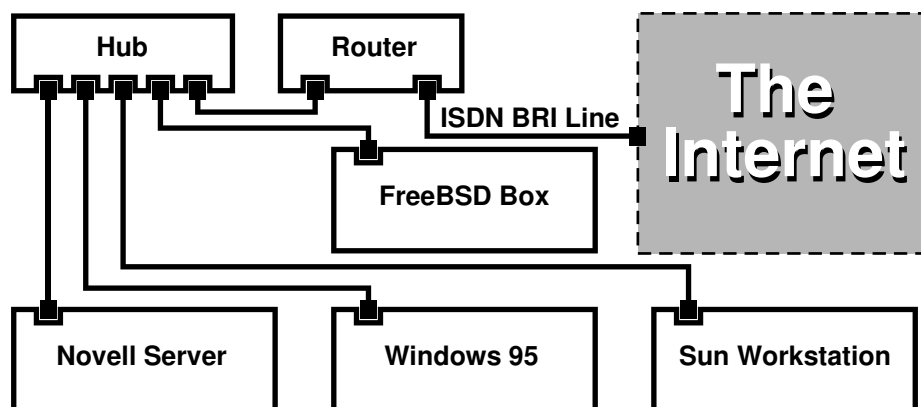
Network uses a bus based topology with 10 base 2 Ethernet („thinnet”). Connect router to network cable with AUI/10BT transceiver, if necessary.



If your home/branch office is only one computer you can use a twisted pair crossover cable to connect to the stand-alone router directly.

Przykład 27-2. Head Office or Other LAN

Network uses a star topology with 10 base T Ethernet („Twisted Pair”).



One large advantage of most routers/bridges is that they allow you to have 2 *separate independent* PPP connections to 2 separate sites at the *same* time. This is not supported on most TA's, except for specific (usually expensive) models that have two serial ports. Do not confuse this with channel bonding, MPP, etc.

This can be a very useful feature if, for example, you have an dedicated ISDN connection at your office and would like to tap into it, but do not want to get another ISDN line at work. A router at the office location can manage a dedicated B channel connection (64 Kbps) to the Internet and use the other B channel for a separate data connection.

The second B channel can be used for dial-in, dial-out or dynamically bonding (MPP, etc.) with the first B channel for more bandwidth.

An Ethernet bridge will also allow you to transmit more than just IP traffic. You can also send IPX/SPX or whatever other protocols you use.

27.8. Network Address Translation

Contributed by Chern Lee.

27.8.1. Overview

FreeBSD's Network Address Translation daemon, commonly known as `natd(8)` is a daemon that accepts incoming raw IP packets, changes the source to the local machine and re-injects these packets back into the outgoing IP packet stream. `natd(8)` does this by changing the source IP address and port such that when data is received back, it is able to determine the original location of the data and forward it back to its original requester.

The most common use of NAT is to perform what is commonly known as Internet Connection Sharing.

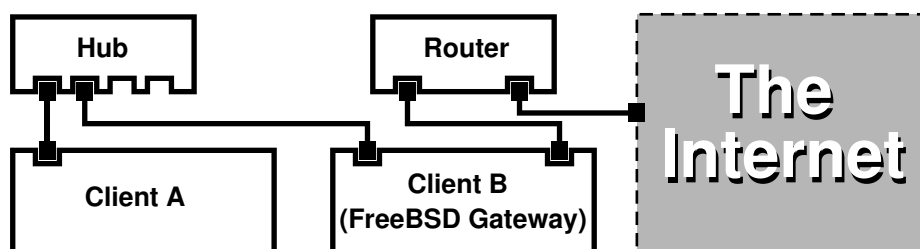
27.8.2. Setup

Due to the diminishing IP space in IPv4, and the increased number of users on high-speed consumer lines such as cable or DSL, people are increasingly in need of an Internet Connection Sharing solution. The ability to connect several computers online through one connection and IP address makes `natd(8)` a reasonable choice.

Most commonly, a user has a machine connected to a cable or DSL line with one IP address and wishes to use this one connected computer to provide Internet access to several more over a LAN.

To do this, the FreeBSD machine on the Internet must act as a gateway. This gateway machine must have two NICs—one for connecting to the Internet router, the other connecting to a LAN. All the machines on the LAN are connected through a hub or switch.

Notatka: There are many ways to get a LAN connected to the Internet through a FreeBSD gateway. This example will only cover a gateway with at least two NICs.



A setup like this is commonly used to share an Internet connection. One of the LAN machines is connected to the Internet. The rest of the machines access the Internet through that „gateway” machine.

27.8.3. Configuration

The following options must be in the kernel configuration file:

```
options IPFIREWALL
options IPDIVERT
```

Additionally, at choice, the following may also be suitable:

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
```

The following must be in `/etc/rc.conf`:

```
gateway_enable="YES" ❶
firewall_enable="YES" ❷
firewall_type="OPEN" ❸
natd_enable="YES"
natd_interface="fxp0" ❹
natd_flags="" ❺
```

- ❶ Sets up the machine to act as a gateway. Running `sysctl net.inet.ip.forwarding=1` would have the same effect.
- ❷ Enables the firewall rules in `/etc/rc.firewall` at boot.
- ❸ This specifies a predefined firewall ruleset that allows anything in. See `/etc/rc.firewall` for additional types.
- ❹ Indicates which interface to forward packets through (the interface connected to the Internet).
- ❺ Any additional configuration options passed to `natd(8)` on boot.

Having the previous options defined in `/etc/rc.conf` would run `natd -interface fxp0` at boot. This can also be run manually.

Notatka: It is also possible to use a configuration file for `natd(8)` when there are too many options to pass. In this case, the configuration file must be defined by adding the following line to `/etc/rc.conf`:

```
natd_flags="-f /etc/natd.conf"
```

The `/etc/natd.conf` file will contain a list of configuration options, one per line. For example the next section case would use the following file:

```
redirect_port tcp 192.168.0.2:6667 6667
redirect_port tcp 192.168.0.3:80 80
```

For more information about the configuration file, consult the `natd(8)` manual page about the `-f` option.

Each machine and interface behind the LAN should be assigned IP address numbers in the private network space as defined by RFC 1918 (<ftp://ftp.isi.edu/in-notes/rfc1918.txt>) and have a default gateway of the **natd** machine's internal IP address.

For example, client A and B behind the LAN have IP addresses of 192.168.0.2 and 192.168.0.3, while the `natd` machine's LAN interface has an IP address of 192.168.0.1. Client A and B's default gateway must be set to that of

the **natd** machine, 192.168.0.1. The **natd** machine's external, or Internet interface does not require any special modification for natd(8) to work.

27.8.4. Port Redirection

The drawback with natd(8) is that the LAN clients are not accessible from the Internet. Clients on the LAN can make outgoing connections to the world but cannot receive incoming ones. This presents a problem if trying to run Internet services on one of the LAN client machines. A simple way around this is to redirect selected Internet ports on the **natd** machine to a LAN client.

For example, an IRC server runs on client A, and a web server runs on client B. For this to work properly, connections received on ports 6667 (IRC) and 80 (web) must be redirected to the respective machines.

The `-redirect_port` must be passed to natd(8) with the proper options. The syntax is as follows:

```
-redirect_port proto targetIP:targetPORT[-targetPORT]
                [aliasIP:]aliasPORT[-aliasPORT]
                [remoteIP[:remotePORT[-remotePORT]]]
```

In the above example, the argument should be:

```
-redirect_port tcp 192.168.0.2:6667 6667
-redirect_port tcp 192.168.0.3:80 80
```

This will redirect the proper *tcp* ports to the LAN client machines.

The `-redirect_port` argument can be used to indicate port ranges over individual ports. For example, *tcp 192.168.0.2:2000-3000 2000-3000* would redirect all connections received on ports 2000 to 3000 to ports 2000 to 3000 on client A.

These options can be used when directly running natd(8), placed within the `natd_flags=""` option in `/etc/rc.conf`, or passed via a configuration file.

For further configuration options, consult natd(8)

27.8.5. Address Redirection

Address redirection is useful if several IP addresses are available, yet they must be on one machine. With this, natd(8) can assign each LAN client its own external IP address. natd(8) then rewrites outgoing packets from the LAN clients with the proper external IP address and redirects all traffic incoming on that particular IP address back to the specific LAN client. This is also known as static NAT. For example, the IP addresses 128.1.1.1, 128.1.1.2, and 128.1.1.3 belong to the **natd** gateway machine. 128.1.1.1 can be used as the **natd** gateway machine's external IP address, while 128.1.1.2 and 128.1.1.3 are forwarded back to LAN clients A and B.

The `-redirect_address` syntax is as follows:

```
-redirect_address localIP publicIP
```

localIP

publicIP

The internal IP address of the LAN client.

The external IP address corresponding to the LAN client.

In the example, this argument would read:

```
-redirect_address 192.168.0.2 128.1.1.2
-redirect_address 192.168.0.3 128.1.1.3
```

Like `-redirect_port`, these arguments are also placed within the `natd_flags=""` option of `/etc/rc.conf`, or passed via a configuration file. With address redirection, there is no need for port redirection since all data received on a particular IP address is redirected.

The external IP addresses on the **natd** machine must be active and aliased to the external interface. Look at `rc.conf(5)` to do so.

27.9. Parallel Line IP (PLIP)

PLIP lets us run TCP/IP between parallel ports. It is useful on machines without network cards, or to install on laptops. In this section, we will discuss:

- Creating a parallel (laplink) cable.
- Connecting two computers with PLIP.

27.9.1. Creating a Parallel Cable

You can purchase a parallel cable at most computer supply stores. If you cannot do that, or you just want to know how it is done, the following table shows how to make one out of a normal parallel printer cable.

Tabela 27-1. Wiring a Parallel Cable for Networking

A-name	A-End	B-End	Descr.	Post/Bit
DATA0	2	15	Data	0/0x01
-ERROR	15	2		1/0x08
DATA1	3	13	Data	0/0x02
+SLCT	13	3		1/0x10
DATA2	4	12	Data	0/0x04
+PE	12	4		1/0x20
DATA3	5	10	Strobe	0/0x08
-ACK	10	5		1/0x40
DATA4	6	11	Data	0/0x10
BUSY	11	6		1/0x80
GND	18-25	18-25	GND	-

27.9.2. Setting Up PLIP

First, you have to get a laplink cable. Then, confirm that both computers have a kernel with `lpt(4)` driver support:

```
# grep lp /var/run/dmesg.boot
```

```
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
```

The parallel port must be an interrupt driven port, you should have lines similar to the following in your in the `/boot/device.hints` file:

```
hint.ppc.0.at="isa"
hint.ppc.0.irq="7"
```

Then check if the kernel configuration file has a `device plip` line or if the `plip.ko` kernel module is loaded. In both cases the parallel networking interface should appear when you use the `ifconfig(8)` command to display it:

```
# ifconfig plip0
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
```

Plug the laplink cable into the parallel interface on both computers.

Configure the network interface parameters on both sites as `root`. For example, if you want to connect the host `host1` with another machine `host2`:

```

                host1 <-----> host2
IP Address      10.0.0.1      10.0.0.2
```

Configure the interface on `host1` by doing:

```
# ifconfig plip0 10.0.0.1 10.0.0.2
```

Configure the interface on `host2` by doing:

```
# ifconfig plip0 10.0.0.2 10.0.0.1
```

You now should have a working connection. Please read the manual pages `lp(4)` and `lpt(4)` for more details.

You should also add both hosts to `/etc/hosts`:

```
127.0.0.1          localhost.my.domain localhost
10.0.0.1           host1.my.domain host1
10.0.0.2           host2.my.domain
```

To confirm the connection works, go to each host and ping the other. For example, on `host1`:

```
# ifconfig plip0
plip0: flags=8851<UP,POINTOPOINT,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.0.0.1 --> 10.0.0.2 netmask 0xff000000
# netstat -r
Routing tables

Internet:
Destination      Gateway          Flags          Refs          Use          Netif Expire
host2            host1           UH              0              0           plip0
# ping -c 4 host2
PING host2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: icmp_seq=0 ttl=255 time=2.774 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=255 time=2.530 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=255 time=2.556 ms
```

```
64 bytes from 10.0.0.2: icmp_seq=3 ttl=255 time=2.714 ms

--- host2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.530/2.643/2.774/0.103 ms
```

27.10. IPv6

Originally Written by Aaron Kaplan. Restructured and Added by Tom Rhodes. Extended by Brad Davis.

IPv6 (also known as IPng „IP next generation”) is the new version of the well known IP protocol (also known as IPv4). Like the other current *BSD systems, FreeBSD includes the KAME IPv6 reference implementation. So your FreeBSD system comes with all you will need to experiment with IPv6. This section focuses on getting IPv6 configured and running.

In the early 1990s, people became aware of the rapidly diminishing address space of IPv4. Given the expansion rate of the Internet there were two major concerns:

- Running out of addresses. Today this is not so much of a concern anymore since RFC1918 private address space (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16) and Network Address Translation (NAT) are being employed.
- Router table entries were getting too large. This is still a concern today.

IPv6 deals with these and many other issues:

- 128 bit address space. In other words theoretically there are 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses available. This means there are approximately $6.67 * 10^{27}$ IPv6 addresses per square meter on our planet.
- Routers will only store network aggregation addresses in their routing tables thus reducing the average space of a routing table to 8192 entries.

There are also lots of other useful features of IPv6 such as:

- Address autoconfiguration (RFC2462 (<http://www.ietf.org/rfc/rfc2462.txt>))
- Anycast addresses („one-out-of many”)
- Mandatory multicast addresses
- IPsec (IP security)
- Simplified header structure
- Mobile IP
- IPv6-to-IPv4 transition mechanisms

For more information see:

- IPv6 overview at playground.sun.com (<http://playground.sun.com/pub/ipng/html/ipng-main.html>)
- KAME.net (<http://www.kame.net>)

- 6bone.net (<http://www.6bone.net>)

27.10.1. Background on IPv6 Addresses

There are different types of IPv6 addresses: Unicast, Anycast and Multicast.

Unicast addresses are the well known addresses. A packet sent to a unicast address arrives exactly at the interface belonging to the address.

Anycast addresses are syntactically indistinguishable from unicast addresses but they address a group of interfaces. The packet destined for an anycast address will arrive at the nearest (in router metric) interface. Anycast addresses may only be used by routers.

Multicast addresses identify a group of interfaces. A packet destined for a multicast address will arrive at all interfaces belonging to the multicast group.

Notatka: The IPv4 broadcast address (usually `xxx.xxx.xxx.255`) is expressed by multicast addresses in IPv6.

Tabela 27-2. Reserved IPv6 addresses

IPv6 address	Prefixlength (Bits)	Description	Notes
::	128 bits	unspecified	cf. 0.0.0.0 in IPv4
::1	128 bits	loopback address	cf. 127.0.0.1 in IPv4
::00:xx:xx:xx:xx	96 bits	embedded IPv4	The lower 32 bits are the IPv4 address. Also called „IPv4 compatible IPv6 address”
::ff:xx:xx:xx:xx	96 bits	IPv4 mapped IPv6 address	The lower 32 bits are the IPv4 address. For hosts which do not support IPv6. cf. loopback address in IPv4
fe80:: - feb::	10 bits	link-local	
fec0:: - fef::	10 bits	site-local	
ff::	8 bits	multicast	
001 (base 2)	3 bits	global unicast	All global unicast addresses are assigned from this pool. The first 3 bits are „001”.

27.10.2. Reading IPv6 Addresses

The canonical form is represented as: `x:x:x:x:x:x:x:x`, each „x” being a 16 Bit hex value. For example `FEBC:A574:382B:23C1:AA49:4592:4EFE:9982`

Often an address will have long substrings of all zeros therefore one such substring per address can be abbreviated by „:”. Also up to three leading „0”s per hexquad can be omitted. For example `fe80::1` corresponds to the canonical form `fe80:0000:0000:0000:0000:0000:0001`.

A third form is to write the last 32 Bit part in the well known (decimal) IPv4 style with dots „.” as separators. For

example `2002::10.0.0.1` corresponds to the (hexadecimal) canonical representation `2002:0000:0000:0000:0000:0a00:0001` which in turn is equivalent to writing `2002::a00:1`.

By now the reader should be able to understand the following:

```
# ifconfig

r10: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    inet 10.0.0.10 netmask 0xffffffff broadcast 10.0.0.255
    inet6 fe80::200:21ff:fe03:8e1%r10 prefixlen 64 scopeid 0x1
    ether 00:00:21:03:08:e1
    media: Ethernet autoselect (100baseTX )
    status: active
```

`fe80::200:21ff:fe03:8e1%r10` is an auto configured link-local address. It is generated from the MAC address as part of the auto configuration.

For further information on the structure of IPv6 addresses see RFC3513 (<http://www.ietf.org/rfc/rfc3513.txt>).

27.10.3. Getting Connected

Currently there are four ways to connect to other IPv6 hosts and networks:

- Join the experimental 6bone
- Getting an IPv6 network from your upstream provider. Talk to your Internet provider for instructions.
- Tunnel via 6-to-4 (RFC3068 (<http://www.ietf.org/rfc/rfc3068.txt>))
- Use the `net/freenet6` port if you are on a dial-up connection.

Here we will talk on how to connect to the 6bone since it currently seems to be the most popular way.

First take a look at the 6bone (<http://www.6bone.net/>) site and find a 6bone connection nearest to you. Write to the responsible person and with a little bit of luck you will be given instructions on how to set up your connection. Usually this involves setting up a GRE (gif) tunnel.

Here is a typical example on setting up a gif(4) tunnel:

```
# ifconfig gif0 create
# ifconfig gif0
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
# ifconfig gif0 tunnel MY_IPV4_ADDR MY_IPV4_REMOTE_TUNNEL_ENDPOINT_ADDR
# ifconfig gif0 inet6 alias MY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR MY_IPv6_REMOTE_TUNNEL_ENDPOINT_ADDR
```

Replace the capitalized words by the information you received from the upstream 6bone node.

This establishes the tunnel. Check if the tunnel is working by `ping6(8)` 'ing `ff02::1%gif0`. You should receive two ping replies.

Notatka: In case you are intrigued by the address `ff02::1%gif0`, this is a multicast address. `%gif0` states that the multicast address at network interface `gif0` is to be used. Since we `ping` a multicast address the other endpoint of the tunnel should reply as well.

By now setting up a route to your 6bone uplink should be rather straightforward:

```
# route add -inet6 default -interface gif0
# ping6 -n MY_UPLINK

# traceroute6 www.jp.FreeBSD.org
(3ffe:505:2008:1:2a0:24ff:fe57:e561) from 3ffe:8060:100::40:2, 30 hops max, 12 byte packets
 1  atnet-meta6  14.147 ms  15.499 ms  24.319 ms
 2  6bone-gw2-ATNET-NT.ipv6.tilab.com  103.408 ms  95.072 ms  *
 3  3ffe:1831:0:ffff::4  138.645 ms  134.437 ms  144.257 ms
 4  3ffe:1810:0:6:290:27ff:fe79:7677  282.975 ms  278.666 ms  292.811 ms
 5  3ffe:1800:0:ff00::4  400.131 ms  396.324 ms  394.769 ms
 6  3ffe:1800:0:3:290:27ff:fe14:cdee  394.712 ms  397.19 ms  394.102 ms
```

This output will differ from machine to machine. By now you should be able to reach the IPv6 site www.kame.net (<http://www.kame.net>) and see the dancing tortoise — that is if you have a IPv6 enabled browser such as www/mozilla, **Konqueror**, which is part of x11/kdebase3, or www/epiphany.

27.10.4. DNS in the IPv6 World

There used to be two types of DNS records for IPv6. The IETF has declared A6 records obsolete. AAAA records are the standard now.

Using AAAA records is straightforward. Assign your hostname to the new IPv6 address you just received by adding:

```
MYHOSTNAME          AAAA      MYIPv6ADDR
```

To your primary zone DNS file. In case you do not serve your own DNS zones ask your DNS provider. Current versions of **bind** (version 8.3 and 9) and **dns/djbdns** (with the IPv6 patch) support AAAA records.

27.10.5. Applying the needed changes to `/etc/rc.conf`

27.10.5.1. IPv6 Client Settings

These settings will help you configure a machine that will be on your LAN and act as a client, not a router. To have **rtsol(8)** autoconfigure your interface on boot all you need to add is:

```
ipv6_enable="YES"
```

To statically assign an IP address such as `2001:471:1f11:251:290:27ff:fee0:2093`, to your `fxp0` interface, add:

```
ipv6_ifconfig_fxp0="2001:471:1f11:251:290:27ff:fee0:2093"
```

To assign a default router of `2001:471:1f11:251::1` add the following to `/etc/rc.conf`:

```
ipv6_defaultrouter="2001:471:1f11:251::1"
```

27.10.5.2. IPv6 Router/Gateway Settings

This will help you take the directions that your tunnel provider, such as the 6bone (<http://www.6bone.net/>), has given you and convert it into settings that will persist through reboots. To restore your tunnel on startup use something like the following in `/etc/rc.conf`:

List the Generic Tunneling interfaces that will be configured, for example `gif0`:

```
gif_interfaces="gif0"
```

To configure the interface with a local endpoint of `MY_IPv4_ADDR` to a remote endpoint of `REMOTE_IPv4_ADDR`:

```
gifconfig_gif0="MY_IPv4_ADDR REMOTE_IPv4_ADDR"
```

To apply the IPv6 address you have been assigned for use as your IPv6 tunnel endpoint, add:

```
ipv6_ifconfig_gif0="MY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR"
```

Then all you have to do is set the default route for IPv6. This is the other side of the IPv6 tunnel:

```
ipv6_defaultrouter="MY_IPv6_REMOTE_TUNNEL_ENDPOINT_ADDR"
```

27.10.5.3. IPv6 Tunnel Settings

If the server is to route IPv6 between the rest of your network and the world, the following `/etc/rc.conf` setting will also be needed:

```
ipv6_gateway_enable="YES"
```

27.10.6. Router Advertisement and Host Auto Configuration

This section will help you setup `rtadvd(8)` to advertise the IPv6 default route.

To enable `rtadvd(8)` you will need the following in your `/etc/rc.conf`:

```
rtadvd_enable="YES"
```

It is important that you specify the interface on which to do IPv6 router solicitation. For example to tell `rtadvd(8)` to use `fxp0`:

```
rtadvd_interfaces="fxp0"
```

Now we must create the configuration file, `/etc/rtadvd.conf`. Here is an example:

```
fxp0:\
    :addrs#1:addr="2001:471:1f11:246::":prefixlen#64:tc=ether:
```

Replace `fxp0` with the interface you are going to be using.

Next, replace `2001:471:1f11:246::` with the prefix of your allocation.

If you are dedicated a /64 subnet you will not need to change anything else. Otherwise, you will need to change the `prefixlen#` to the correct value.

27.11. Asynchronous Transfer Mode (ATM)

Contributed by Harti Brandt.

27.11.1. Configuring classical IP over ATM (PVCs)

Classical IP over ATM (CLIP) is the simplest method to use Asynchronous Transfer Mode (ATM) with IP. It can be used with switched connections (SVCs) and with permanent connections (PVCs). This section describes how to set up a network based on PVCs.

27.11.1.1. Fully meshed configurations

The first method to set up a CLIP with PVCs is to connect each machine to each other machine in the network via a dedicated PVC. While this is simple to configure it tends to become impractical for a larger number of machines. The example supposes that we have four machines in the network, each connected to the ATM network with an ATM adapter card. The first step is the planning of the IP addresses and the ATM connections between the machines. We use the following:

Host	IP Address
hostA	192.168.173.1
hostB	192.168.173.2
hostC	192.168.173.3
hostD	192.168.173.4

To build a fully meshed net we need one ATM connection between each pair of machines:

Machines	VPI.VCI couple
hostA - hostB	0.100
hostA - hostC	0.101
hostA - hostD	0.102
hostB - hostC	0.103
hostB - hostD	0.104
hostC - hostD	0.105

The VPI and VCI values at each end of the connection may of course differ, but for simplicity we assume that they are the same. Next we need to configure the ATM interfaces on each host:

```
hostA# ifconfig hatm0 192.168.173.1 up
hostB# ifconfig hatm0 192.168.173.2 up
hostC# ifconfig hatm0 192.168.173.3 up
hostD# ifconfig hatm0 192.168.173.4 up
```

assuming that the ATM interface is `hatm0` on all hosts. Now the PVCs need to be configured on `hostA` (we assume that they are already configured on the ATM switches, you need to consult the manual for the switch on how to do this).

```
hostA# atmconfig natm add 192.168.173.2 hatm0 0 100 llc/snap ubr
hostA# atmconfig natm add 192.168.173.3 hatm0 0 101 llc/snap ubr
```

```

hostA# atmconfig natm add 192.168.173.4 hatm0 0 102 llc/snap ubr

hostB# atmconfig natm add 192.168.173.1 hatm0 0 100 llc/snap ubr
hostB# atmconfig natm add 192.168.173.3 hatm0 0 103 llc/snap ubr
hostB# atmconfig natm add 192.168.173.4 hatm0 0 104 llc/snap ubr

hostC# atmconfig natm add 192.168.173.1 hatm0 0 101 llc/snap ubr
hostC# atmconfig natm add 192.168.173.2 hatm0 0 103 llc/snap ubr
hostC# atmconfig natm add 192.168.173.4 hatm0 0 105 llc/snap ubr

hostD# atmconfig natm add 192.168.173.1 hatm0 0 102 llc/snap ubr
hostD# atmconfig natm add 192.168.173.2 hatm0 0 104 llc/snap ubr
hostD# atmconfig natm add 192.168.173.3 hatm0 0 105 llc/snap ubr

```

Of course other traffic contracts than UBR can be used given the ATM adapter supports those. In this case the name of the traffic contract is followed by the parameters of the traffic. Help for the atmconfig(8) tool can be obtained with:

```
# atmconfig help natm add
```

or in the atmconfig(8) manual page.

The same configuration can also be done via `/etc/rc.conf`. For hostA this would look like:

```

network_interfaces="lo0 hatm0"
ifconfig_hatm0="inet 192.168.173.1 up"
natm_static_routes="hostB hostC hostD"
route_hostB="192.168.173.2 hatm0 0 100 llc/snap ubr"
route_hostC="192.168.173.3 hatm0 0 101 llc/snap ubr"
route_hostD="192.168.173.4 hatm0 0 102 llc/snap ubr"

```

The current state of all CLIP routes can be obtained with:

```
hostA# atmconfig natm show
```

V. Dodatki

Dodatek A. Obtaining FreeBSD

A.1. CDROM and DVD Publishers

A.1.1. Retail Boxed Products

FreeBSD is available as a boxed product (FreeBSD CDs, additional software, and printed documentation) from several retailers:

- CompUSA
WWW: <http://www.compusa.com/>
- Frys Electronics
WWW: <http://www.frys.com/>

A.1.2. CD and DVD Sets

FreeBSD CD and DVD sets are available from many online retailers:

- BSD Mall by Daemon News
PO Box 161
Nauvoo, IL 62354
USA
Phone: +1 866 273-6255
Fax: +1 217 453-9956
Email: <sales@bsdmail.com>
WWW: <http://www.bsdmail.com/freebsd1.html>
- BSD-Systems
Email: <info@bsd-systems.co.uk>
WWW: <http://www.bsd-systems.co.uk>
- FreeBSD Mall, Inc.
3623 Sanford Street
Concord, CA 94520-1405
USA
Phone: +1 925 674-0783
Fax: +1 925 674-0821
Email: <info@freebsdmail.com>
WWW: <http://www.freebsdmail.com/>
- Hinner EDV
St. Augustinus-Str. 10
D-81825 München
Germany
Phone: (089) 428 419
WWW: <http://www.hinner.de/linux/freebsd.html>

- Ikarios
22-24 rue Voltaire
92000 Nanterre
France
WWW: <http://ikarios.com/form/#freebsd>
- JMC Software
Ireland
Phone: 353 1 6291282
WWW: <http://www.thelinuxmall.com>
- Linux CD Mall
Private Bag MBE N348
Auckland 1030
New Zealand
Phone: +64 21 866529
WWW: <http://www.linuxcdmall.co.nz/>
- The Linux Emporium
Hilliard House, Lester Way
Wallingford
OX10 9TA
United Kingdom
Phone: +44 1491 837010
Fax: +44 1491 837016
WWW: <http://www.linuxemporium.co.uk/products/freebsd/>
- Linux+ DVD Magazine
Lewartowskiego 6
Warsaw
00-190
Poland
Phone: +48 22 860 18 18
Email: [<editors@lpmagazine.org>](mailto:editors@lpmagazine.org)
WWW: <http://www.lpmagazine.org/>
- Linux System Labs Australia
21 Ray Drive
Balwyn North
VIC - 3104
Australia
Phone: +61 3 9857 5918
Fax: +61 3 9857 8974
WWW: <http://www.lsl.com.au>
- LinuxCenter.Ru
Galernaya Street, 55
Saint-Petersburg
190000
Russia
Phone: +7-812-3125208

Email: <info@linuxcenter.ru>
WWW: <http://linuxcenter.ru/freebsd>

A.1.3. Distributors

If you are a reseller and want to carry FreeBSD CDROM products, please contact a distributor:

- Cylogistics
809B Cuesta Dr., #2149
Mountain View, CA 94040
USA
Phone: +1 650 694-4949
Fax: +1 650 694-4953
Email: <sales@cylogistics.com>
WWW: <http://www.cylogistics.com/>
- Ingram Micro
1600 E. St. Andrew Place
Santa Ana, CA 92705-4926
USA
Phone: 1 (800) 456-8000
WWW: <http://www.ingrammicro.com/>
- Kudzu, LLC
7375 Washington Ave. S.
Edina, MN 55439
USA
Phone: +1 952 947-0822
Fax: +1 952 947-0876
Email: <sales@kudzuenterpises.com>
- LinuxCenter.Ru
Galernaya Street, 55
Saint-Petersburg
190000
Russia
Phone: +7-812-3125208
Email: <info@linuxcenter.ru>
WWW: <http://linuxcenter.ru/freebsd>
- Navarre Corp
7400 49th Ave South
New Hope, MN 55428
USA
Phone: +1 763 535-8333
Fax: +1 763 535-0341
WWW: <http://www.navarre.com/>

A.2. FTP Sites

The official sources for FreeBSD are available via anonymous FTP from a worldwide set of mirror sites. The site <ftp://ftp.FreeBSD.org/pub/FreeBSD/> is well connected and allows a large number of connections to it, but you are probably better off finding a „closer” mirror site (especially if you decide to set up some sort of mirror site).

The FreeBSD mirror sites database (<http://mirrorlist.FreeBSD.org/>) is more accurate than the mirror listing in the Handbook, as it gets its information from the DNS rather than relying on static lists of hosts.

Additionally, FreeBSD is available via anonymous FTP from the following mirror sites. If you choose to obtain FreeBSD via anonymous FTP, please try to use a site near you. The mirror sites listed as „Primary Mirror Sites” typically have the entire FreeBSD archive (all the currently available versions for each of the architectures) but you will probably have faster download times from a site that is in your country or region. The regional sites carry the most recent versions for the most popular architecture(s) but might not carry the entire FreeBSD archive. All sites provide access via anonymous FTP but some sites also provide access via other methods. The access methods available for each site are provided in parentheses after the hostname.

Central Servers, Primary Mirror Sites, Armenia, Australia, Austria, Brazil, Canada, China, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hong Kong, Hungary, Iceland, Ireland, Israel, Italy, Japan, Korea, Latvia, Lithuania, Netherlands, New Zealand, Norway, Poland, Portugal, Romania, Russia, Saudi Arabia, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Turkey, Ukraine, United Kingdom, USA.

(as of UTC)

Central Servers

- <ftp://ftp.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp.FreeBSD.org/pub/FreeBSD/>))

Primary Mirror Sites

In case of problems, please contact the hostmaster <mirror-admin@FreeBSD.org> for this domain.

- <ftp://ftp1.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp4.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp4.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp5.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp9.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp10.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp10.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp10.FreeBSD.org/pub/FreeBSD/>))

- <ftp://ftp11.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp12.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp13.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp14.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp14.FreeBSD.org/pub/FreeBSD/>))

Armenia

In case of problems, please contact the hostmaster <hostmaster@am.FreeBSD.org> for this domain.

- <ftp://ftp1.am.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp1.am.FreeBSD.org/pub/FreeBSD/>)) / rsync)

Australia

In case of problems, please contact the hostmaster <hostmaster@au.FreeBSD.org> for this domain.

- <ftp://ftp.au.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.au.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.au.FreeBSD.org/pub/FreeBSD/> (ftp)

Austria

In case of problems, please contact the hostmaster <hostmaster@at.FreeBSD.org> for this domain.

- <ftp://ftp.at.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp.at.FreeBSD.org/pub/FreeBSD/>)) / httpv6 (<http://ftp.at.FreeBSD.org/pub/FreeBSD/>)

Brazil

In case of problems, please contact the hostmaster <hostmaster@br.FreeBSD.org> for this domain.

- <ftp://ftp.br.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.br.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp2.br.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp2.br.FreeBSD.org/>))
- <ftp://ftp3.br.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp4.br.FreeBSD.org/pub/FreeBSD/> (ftp)
- ftp://ftp5.br.FreeBSD.org
- <ftp://ftp6.br.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.br.FreeBSD.org/pub/FreeBSD/> (ftp)

Canada

In case of problems, please contact the hostmaster <hostmaster@ca.FreeBSD.org> for this domain.

- <ftp://ftp.ca.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.ca.FreeBSD.org/pub/FreeBSD/> (ftp)

- <ftp://ftp3.ca.FreeBSD.org/pub/FreeBSD/> (ftp)

China

In case of problems, please contact the hostmaster <hostmaster@cn.FreeBSD.org> for this domain.

- <ftp://ftp.cn.FreeBSD.org/pub/FreeBSD/> (ftp)

Czech Republic

In case of problems, please contact the hostmaster <hostmaster@cz.FreeBSD.org> for this domain.

- <ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/>) / [http](http://ftp.cz.FreeBSD.org/pub/FreeBSD/) (<http://ftp.cz.FreeBSD.org/pub/FreeBSD/>) / [httpv6](http://ftp.cz.FreeBSD.org/pub/FreeBSD/) (<http://ftp.cz.FreeBSD.org/pub/FreeBSD/>) / rsync / rsyncv6)
- <ftp://ftp2.cz.FreeBSD.org/pub/FreeBSD/> (ftp / [http](http://ftp2.cz.FreeBSD.org/pub/FreeBSD/) (<http://ftp2.cz.FreeBSD.org/pub/FreeBSD/>))

Denmark

In case of problems, please contact the hostmaster <hostmaster@dk.FreeBSD.org> for this domain.

- <ftp://ftp.dk.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / [http](http://ftp.dk.FreeBSD.org/pub/FreeBSD/) (<http://ftp.dk.FreeBSD.org/pub/FreeBSD/>) / [httpv6](http://ftp.dk.FreeBSD.org/pub/FreeBSD/) (<http://ftp.dk.FreeBSD.org/pub/FreeBSD/>))

Estonia

In case of problems, please contact the hostmaster <hostmaster@ee.FreeBSD.org> for this domain.

- <ftp://ftp.ee.FreeBSD.org/pub/FreeBSD/> (ftp)

Finland

In case of problems, please contact the hostmaster <hostmaster@fi.FreeBSD.org> for this domain.

- <ftp://ftp.fi.FreeBSD.org/pub/FreeBSD/> (ftp)

France

In case of problems, please contact the hostmaster <hostmaster@fr.FreeBSD.org> for this domain.

- <ftp://ftp.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp1.fr.FreeBSD.org/pub/FreeBSD/> (ftp / [http](http://ftp1.fr.FreeBSD.org/pub/FreeBSD/) (<http://ftp1.fr.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp2.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.fr.FreeBSD.org/pub/FreeBSD/> (ftp)

- <ftp://ftp4.fr.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp4.fr.FreeBSD.org/pub/FreeBSD/>) / [http](http://ftp4.fr.FreeBSD.org/pub/FreeBSD/) (<http://ftp4.fr.FreeBSD.org/pub/FreeBSD/>) / [httpv6](http://ftp4.fr.FreeBSD.org/pub/FreeBSD/) (<http://ftp4.fr.FreeBSD.org/pub/FreeBSD/>) / [rsync](rsync://ftp4.fr.FreeBSD.org/FreeBSD/) (<rsync://ftp4.fr.FreeBSD.org/FreeBSD/>) / [rsyncv6](rsync://ftp4.fr.FreeBSD.org/FreeBSD/) (<rsync://ftp4.fr.FreeBSD.org/FreeBSD/>))
- <ftp://ftp5.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.fr.FreeBSD.org/pub/FreeBSD/> (ftp / [rsync](rsync://ftp6.fr.FreeBSD.org/FreeBSD/))
- <ftp://ftp7.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.fr.FreeBSD.org/pub/FreeBSD/> (ftp)

Germany

In case of problems, please contact the hostmaster <de-bsd-hubs@de.FreeBSD.org> for this domain.

- <ftp://ftp.de.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp1.de.FreeBSD.org/freebsd/> (ftp / [http](http://www1.de.FreeBSD.org/freebsd/) (<http://www1.de.FreeBSD.org/freebsd/>) / [rsync](rsync://rsync3.de.FreeBSD.org/freebsd/) (<rsync://rsync3.de.FreeBSD.org/freebsd/>))
- <ftp://ftp2.de.FreeBSD.org/pub/FreeBSD/> (ftp / [http](http://ftp2.de.FreeBSD.org/pub/FreeBSD/) (<http://ftp2.de.FreeBSD.org/pub/FreeBSD/>) / [rsync](rsync://ftp2.de.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp3.de.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.de.FreeBSD.org/FreeBSD/> (ftp / [http](http://ftp4.de.FreeBSD.org/pub/FreeBSD/) (<http://ftp4.de.FreeBSD.org/pub/FreeBSD/>) / [rsync](rsync://ftp4.de.FreeBSD.org/FreeBSD/))
- <ftp://ftp5.de.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.de.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.de.FreeBSD.org/pub/FreeBSD/> (ftp / [http](http://ftp7.de.FreeBSD.org/pub/FreeBSD/) (<http://ftp7.de.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp8.de.FreeBSD.org/pub/FreeBSD/> (ftp)

Greece

In case of problems, please contact the hostmaster <hostmaster@gr.FreeBSD.org> for this domain.

- <ftp://ftp.gr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.gr.FreeBSD.org/pub/FreeBSD/> (ftp)

Hong Kong

- <ftp://ftp.hk.FreeBSD.org/pub/FreeBSD/> (ftp)

Hungary

In case of problems, please contact the hostmaster <hostmaster@hu.FreeBSD.org> for this domain.

- <ftp://ftp.hu.FreeBSD.org/pub/FreeBSD/> (ftp / [http](http://ftp.hu.FreeBSD.org/pub/FreeBSD/) (<http://ftp.hu.FreeBSD.org/pub/FreeBSD/>) / [rsync](rsync://ftp.hu.FreeBSD.org/FreeBSD/))

Iceland

In case of problems, please contact the hostmaster <hostmaster@is.FreeBSD.org> for this domain.

- <ftp://ftp.is.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)

Ireland

In case of problems, please contact the hostmaster <hostmaster@ie.FreeBSD.org> for this domain.

- <ftp://ftp.ie.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.ie.FreeBSD.org/pub/FreeBSD/> (ftp / [http \(http://ftp2.ie.FreeBSD.org/pub/FreeBSD/\) / rsync](http://ftp2.ie.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp3.ie.FreeBSD.org/pub/FreeBSD/> (ftp / [http \(http://ftp3.ie.FreeBSD.org/pub/FreeBSD/\) / rsync](http://ftp3.ie.FreeBSD.org/pub/FreeBSD/))

Israel

In case of problems, please contact the hostmaster <hostmaster@il.FreeBSD.org> for this domain.

- <ftp://ftp.il.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6)

Italy

In case of problems, please contact the hostmaster <hostmaster@it.FreeBSD.org> for this domain.

- <ftp://ftp.it.FreeBSD.org/pub/FreeBSD/> (ftp)

Japan

In case of problems, please contact the hostmaster <hostmaster@jp.FreeBSD.org> for this domain.

- <ftp://ftp.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp9.jp.FreeBSD.org/pub/FreeBSD/> (ftp)

Korea

In case of problems, please contact the hostmaster <hostmaster@kr.FreeBSD.org> for this domain.

- <ftp://ftp.kr.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)

- <ftp://ftp2.kr.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp2.kr.FreeBSD.org/pub/FreeBSD/>))

Latvia

In case of problems, please contact the hostmaster <hostmaster@lv.FreeBSD.org> for this domain.

- <ftp://ftp.lv.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.lv.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp2.lv.FreeBSD.org/pub/FreeBSD/> (ftp)

Lithuania

In case of problems, please contact the hostmaster <hostmaster@lt.FreeBSD.org> for this domain.

- <ftp://ftp.lt.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.lt.FreeBSD.org/pub/FreeBSD/>))

Netherlands

In case of problems, please contact the hostmaster <hostmaster@nl.FreeBSD.org> for this domain.

- <ftp://ftp.nl.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.nl.FreeBSD.org/os/FreeBSD/>) / rsync)
- <ftp://ftp2.nl.FreeBSD.org/pub/FreeBSD/> (ftp)

New Zealand

- <ftp://ftp.nz.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.nz.FreeBSD.org/pub/FreeBSD/>))

Norway

In case of problems, please contact the hostmaster <hostmaster@no.FreeBSD.org> for this domain.

- <ftp://ftp.no.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp3.no.FreeBSD.org/pub/FreeBSD/> (ftp)

Poland

In case of problems, please contact the hostmaster <hostmaster@pl.FreeBSD.org> for this domain.

- <ftp://ftp.pl.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.pl.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<http://ftp2.pl.FreeBSD.org/pub/FreeBSD/>) / http (<http://ftp2.pl.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp2.pl.FreeBSD.org/pub/FreeBSD/>) / rsync / rsyncv6)

Portugal

In case of problems, please contact the hostmaster <hostmaster@pt.FreeBSD.org> for this domain.

- <ftp://ftp.pt.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.pt.FreeBSD.org/pub/freebsd/> (ftp)
- <ftp://ftp4.pt.FreeBSD.org/pub/ISO/FreeBSD/> (ftp)

Romania

In case of problems, please contact the hostmaster <hostmaster@ro.FreeBSD.org> for this domain.

- <ftp://ftp.ro.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp1.ro.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / http (<http://ftp1.ro.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp1.ro.FreeBSD.org/pub/FreeBSD/>))

Russia

In case of problems, please contact the hostmaster <hostmaster@ru.FreeBSD.org> for this domain.

- <ftp://ftp.ru.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.ru.FreeBSD.org/FreeBSD/>) / rsync)
- <ftp://ftp2.ru.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp2.ru.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp3.ru.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.ru.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.ru.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp5.ru.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp6.ru.FreeBSD.org/pub/FreeBSD/> (ftp)

Saudi Arabia

In case of problems, please contact the hostmaster <ftpadmin@isu.net.sa> for this domain.

- <ftp://ftp.isu.net.sa/pub/ftp.freebsd.org/> (ftp)

Slovak Republic

In case of problems, please contact the hostmaster <hostmaster@sk.FreeBSD.org> for this domain.

- <ftp://ftp.sk.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp.sk.FreeBSD.org/pub/FreeBSD/>) / http (<http://ftp.sk.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp.sk.FreeBSD.org/pub/FreeBSD/>) / rsync / rsyncv6)
- <ftp://ftp2.sk.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp2.sk.FreeBSD.org/pub/FreeBSD/>) / http (<http://ftp2.sk.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp2.sk.FreeBSD.org/pub/FreeBSD/>))

Slovenia

In case of problems, please contact the hostmaster <hostmaster@si.FreeBSD.org> for this domain.

- <ftp://ftp.si.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.si.FreeBSD.org/pub/FreeBSD/> (ftp)

South Africa

In case of problems, please contact the hostmaster <hostmaster@za.FreeBSD.org> for this domain.

- <ftp://ftp.za.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.za.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.za.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.za.FreeBSD.org/pub/FreeBSD/> (ftp)

Spain

In case of problems, please contact the hostmaster <hostmaster@es.FreeBSD.org> for this domain.

- <ftp://ftp.es.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.es.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp2.es.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.es.FreeBSD.org/pub/FreeBSD/> (ftp)

Sweden

In case of problems, please contact the hostmaster <hostmaster@se.FreeBSD.org> for this domain.

- <ftp://ftp.se.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.se.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp2.se.FreeBSD.org/>) / rsync (<rsync://ftp2.se.FreeBSD.org/>))
- <ftp://ftp3.se.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/>) / http (<http://ftp4.se.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp4.se.FreeBSD.org/pub/FreeBSD/>) / rsync (<rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/>) / rsyncv6 (<rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp5.se.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp5.se.FreeBSD.org/>) / rsync)

Switzerland

In case of problems, please contact the hostmaster <hostmaster@ch.FreeBSD.org> for this domain.

- <ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.ch.FreeBSD.org/pub/FreeBSD/>))

Taiwan

In case of problems, please contact the hostmaster <hostmaster@tw.FreeBSD.org> for this domain.

- <ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/>) / rsync / rsyncv6)
- <ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 (<ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/>) / http (<http://ftp2.tw.FreeBSD.org/pub/FreeBSD/>) / httpv6 (<http://ftp2.tw.FreeBSD.org/pub/FreeBSD/>) / rsync / rsyncv6)
- <ftp://ftp3.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.tw.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp6.tw.FreeBSD.org/>) / rsync)
- <ftp://ftp7.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp9.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp10.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp11.tw.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp11.tw.FreeBSD.org/FreeBSD/>))
- <ftp://ftp12.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp13.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp14.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp15.tw.FreeBSD.org/pub/FreeBSD/> (ftp)

Turkey

- <ftp://ftp.tr.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.tr.FreeBSD.org/pub/FreeBSD/>) / rsync)
- <ftp://ftp2.tr.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)

Ukraine

- <ftp://ftp.ua.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp.ua.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp2.ua.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp2.ua.FreeBSD.org/pub/FreeBSD/>))
- <ftp://ftp7.ua.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.ua.FreeBSD.org/pub/FreeBSD/> (ftp / http (<http://ftp8.ua.FreeBSD.org/FreeBSD/>))
- <ftp://ftp11.ua.FreeBSD.org/pub/FreeBSD/> (ftp)

United Kingdom

In case of problems, please contact the hostmaster <hostmaster@uk.FreeBSD.org> for this domain.

- <ftp://ftp.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.uk.FreeBSD.org/pub/FreeBSD/> (ftp / [http \(http://ftp2.uk.FreeBSD.org/\) / rsync](http://ftp2.uk.FreeBSD.org/))
- <ftp://ftp3.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.uk.FreeBSD.org/pub/FreeBSD/> (ftp)

USA

In case of problems, please contact the hostmaster <hostmaster@us.FreeBSD.org> for this domain.

- <ftp://ftp1.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.us.FreeBSD.org/pub/FreeBSD/> (ftp / [http \(http://ftp4.us.FreeBSD.org/pub/FreeBSD/\) /](http://ftp4.us.FreeBSD.org/pub/FreeBSD/) [httpv6 \(http://ftp4.us.FreeBSD.org/pub/FreeBSD/\)](http://ftp4.us.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp5.us.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp6.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.us.FreeBSD.org/pub/FreeBSD/> (ftp / [http \(http://ftp7.us.FreeBSD.org/pub/FreeBSD/\) / rsync](http://ftp7.us.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp8.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp9.us.FreeBSD.org/pub/FreeBSD/> (ftp / [http \(http://ftp9.us.FreeBSD.org/pub/os/FreeBSD/\)](http://ftp9.us.FreeBSD.org/pub/os/FreeBSD/))
- <ftp://ftp10.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp11.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp12.us.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp13.us.FreeBSD.org/pub/FreeBSD/> (ftp / [http \(http://ftp13.us.FreeBSD.org/pub/FreeBSD/\) / rsync](http://ftp13.us.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp14.us.FreeBSD.org/pub/FreeBSD/> (ftp / [http \(http://ftp14.us.FreeBSD.org/pub/FreeBSD/\)](http://ftp14.us.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp15.us.FreeBSD.org/pub/FreeBSD/> (ftp)

A.3. Anonymous CVS

A.3.1. Introduction

Anonymous CVS (or, as it is otherwise known, *anoncvs*) is a feature provided by the CVS utilities bundled with FreeBSD for synchronizing with a remote CVS repository. Among other things, it allows users of FreeBSD to perform, with no special privileges, read-only CVS operations against one of the FreeBSD project's official anoncvs

servers. To use it, one simply sets the `CVSROOT` environment variable to point at the appropriate `anoncvs` server, provides the well-known password „anoncvs” with the `cvs login` command, and then uses the `cvs(1)` command to access it like any local repository.

Notatka: The `cvs login` command, stores the passwords that are used for authenticating to the CVS server in a file called `.cvspass` in your `HOME` directory. If this file does not exist, you might get an error when trying to use `cvs login` for the first time. Just make an empty `.cvspass` file, and retry to login.

While it can also be said that the **CVSup** and *anoncvs* services both perform essentially the same function, there are various trade-offs which can influence the user’s choice of synchronization methods. In a nutshell, **CVSup** is much more efficient in its usage of network resources and is by far the most technically sophisticated of the two, but at a price. To use **CVSup**, a special client must first be installed and configured before any bits can be grabbed, and then only in the fairly large chunks which **CVSup** calls *collections*.

Anoncvs, by contrast, can be used to examine anything from an individual file to a specific program (like `ls` or `grep`) by referencing the CVS module name. Of course, **anoncvs** is also only good for read-only operations on the CVS repository, so if it is your intention to support local development in one repository shared with the FreeBSD project bits then **CVSup** is really your only option.

A.3.2. Using Anonymous CVS

Configuring `cvs(1)` to use an Anonymous CVS repository is a simple matter of setting the `CVSROOT` environment variable to point to one of the FreeBSD project’s *anoncvs* servers. At the time of this writing, the following servers are available:

- *Austria*: `:pserver:anoncvs@anoncvs.at.FreeBSD.org:/home/ncvs` (Use `cvs login` and enter any password when prompted.)
- *France*: `:pserver:anoncvs@anoncvs.fr.FreeBSD.org:/home/ncvs` (`pserver` (password „anoncvs”), `ssh` (no password))
- *Germany*: `:pserver:anoncvs@anoncvs.de.FreeBSD.org:/home/ncvs` (Use `cvs login` and enter the password „anoncvs” when prompted.)
- *Germany*: `:pserver:anoncvs@anoncvs2.de.FreeBSD.org:/home/ncvs` (`rsh`, `pserver`, `ssh`, `ssh/2022`)
- *Japan*: `:pserver:anoncvs@anoncvs.jp.FreeBSD.org:/home/ncvs` (Use `cvs login` and enter the password „anoncvs” when prompted.)
- *USA*: `freebsdanoncvs@anoncvs.FreeBSD.org:/home/ncvs` (`ssh` only - no password)
 SSH HostKey: 1024 a1:e7:46:de:fb:56:ef:05:bc:73:aa:91:09:da:f7:f4 root@sanmateo.ecn.purdue.edu
 SSH2 HostKey: 1024 52:02:38:1a:2f:a8:71:d3:f5:83:93:8d:aa:00:6f:65 ssh_host_dsa_key.pub
- *USA*: `anoncvs@anoncvs1.FreeBSD.org:/home/ncvs` (`ssh` only - no password)
 SSH HostKey: 1024 8b:c4:6f:9a:7e:65:8a:eb:50:50:29:7c:a1:47:03:bc root@ender.liquidneon.com
 SSH2 HostKey: 2048 4d:59:19:7b:ea:9b:76:0b:ca:ee:da:26:e2:3a:83:b8 ssh_host_dsa_key.pub

Since CVS allows one to „check out” virtually any version of the FreeBSD sources that ever existed (or, in some cases, will exist), you need to be familiar with the revision (`-r`) flag to `cvs(1)` and what some of the permissible values for it in the FreeBSD Project repository are.

There are two kinds of tags, revision tags and branch tags. A revision tag refers to a specific revision. Its meaning stays the same from day to day. A branch tag, on the other hand, refers to the latest revision on a given line of development, at any given time. Because a branch tag does not refer to a specific revision, it may mean something different tomorrow than it means today.

Sekcja A.7 contains revision tags that users might be interested in. Again, none of these are valid for the Ports Collection since the Ports Collection does not have multiple revisions.

When you specify a branch tag, you normally receive the latest versions of the files on that line of development. If you wish to receive some past version, you can do so by specifying a date with the `-D date` flag. See the `cvs(1)` manual page for more details.

A.3.3. Examples

While it really is recommended that you read the manual page for `cvs(1)` thoroughly before doing anything, here are some quick examples which essentially show how to use Anonymous CVS:

Przykład A-1. Checking Out Something from -CURRENT (ls(1)):

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.jp.FreeBSD.org:/home/ncvs
% cvs login
At the prompt, enter the password „anoncvs”.
% cvs co ls
```

Przykład A-2. Using SSH to check out the `src/` tree:

```
% cvs -d freebsdanoncvs@anoncvs.FreeBSD.org:/home/ncvs co src
The authenticity of host 'anoncvs.freebsd.org (128.46.156.46)' can't be established.
DSA key fingerprint is 52:02:38:1a:2f:a8:71:d3:f5:83:93:8d:aa:00:6f:65.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'anoncvs.freebsd.org' (DSA) to the list of known hosts.
```

Przykład A-3. Checking Out the Version of `ls(1)` in the 6-STABLE Branch:

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.jp.FreeBSD.org:/home/ncvs
% cvs login
At the prompt, enter the password „anoncvs”.
% cvs co -rRELEASE_6 ls
```

Przykład A-4. Creating a List of Changes (as Unified Diffs) to `ls(1)`

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.jp.FreeBSD.org:/home/ncvs
% cvs login
At the prompt, enter the password „anoncvs”.
% cvs rdiff -u -rRELEASE_5_3_0_RELEASE -rRELEASE_5_4_0_RELEASE ls
```

Przykład A-5. Finding Out What Other Module Names Can Be Used:

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.jp.FreeBSD.org:/home/ncvs
% cvs login
At the prompt, enter the password „anoncvs”.
% cvs co modules
% more modules/modules
```

A.3.4. Other Resources

The following additional resources may be helpful in learning CVS:

- CVS Tutorial (<http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/cvs/>) from Cal Poly.
- CVS Home (<http://ximbiot.com/cvs/wiki/>), the CVS development and support community.
- CVSweb (<http://www.FreeBSD.org/cgi/cvsweb.cgi>) is the FreeBSD Project web interface for CVS.

A.4. Using CTM

CTM is a method for keeping a remote directory tree in sync with a central one. It has been developed for usage with FreeBSD's source trees, though other people may find it useful for other purposes as time goes by. Little, if any, documentation currently exists at this time on the process of creating deltas, so contact the **ctm-users** (<http://lists.FreeBSD.org/mailman/listinfo/ctm-users>) mailing list for more information and if you wish to use **CTM** for other things.

A.4.1. Why Should I Use CTM?

CTM will give you a local copy of the FreeBSD source trees. There are a number of „flavors” of the tree available. Whether you wish to track the entire CVS tree or just one of the branches, **CTM** can provide you the information. If you are an active developer on FreeBSD, but have lousy or non-existent TCP/IP connectivity, or simply wish to have the changes automatically sent to you, **CTM** was made for you. You will need to obtain up to three deltas per day for the most active branches. However, you should consider having them sent by automatic email. The sizes of the updates are always kept as small as possible. This is typically less than 5K, with an occasional (one in ten) being 10-50K and every now and then a large 100K+ or more coming around.

You will also need to make yourself aware of the various caveats related to working directly from the development sources rather than a pre-packaged release. This is particularly true if you choose the „current” sources. It is recommended that you read *Staying current with FreeBSD*.

A.4.2. What Do I Need to Use CTM?

You will need two things: The **CTM** program, and the initial deltas to feed it (to get up to „current” levels).

The **CTM** program has been part of FreeBSD ever since version 2.0 was released, and lives in `/usr/src/usr.sbin/ctm` if you have a copy of the source available.

The „deltas” you feed **CTM** can be had two ways, FTP or email. If you have general FTP access to the Internet then the following FTP sites support access to **CTM**:

`ftp://ftp.FreeBSD.org/pub/FreeBSD/CTM/`

or see section mirrors.

FTP the relevant directory and fetch the `README` file, starting from there.

If you wish to get your deltas via email:

Subscribe to one of the **CTM** distribution lists. `ctm-cvs-cur` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-cvs-cur>) supports the entire CVS tree. `ctm-src-cur` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-src-cur>) supports the head of the development branch. `ctm-src-4` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-src-4>) supports the 4.X release branch, etc.. (If you do not know how to subscribe yourself to a list, click on the list name above or go to <http://lists.FreeBSD.org/mailman/listinfo> and click on the list that you wish to subscribe to. The list page should contain all of the necessary subscription instructions.)

When you begin receiving your **CTM** updates in the mail, you may use the `ctm_rmail` program to unpack and apply them. You can actually use the `ctm_rmail` program directly from a entry in `/etc/aliases` if you want to have the process run in a fully automated fashion. Check the `ctm_rmail` manual page for more details.

Notatka: No matter what method you use to get the **CTM** deltas, you should subscribe to the `ctm-announce` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-announce>) mailing list. In the future, this will be the only place where announcements concerning the operations of the **CTM** system will be posted. Click on the list name above and follow the instructions to subscribe to the list.

A.4.3. Using CTM for the First Time

Before you can start using **CTM** deltas, you will need to get to a starting point for the deltas produced subsequently to it.

First you should determine what you already have. Everyone can start from an „empty” directory. You must use an initial „Empty” delta to start off your **CTM** supported tree. At some point it is intended that one of these „started” deltas be distributed on the CD for your convenience, however, this does not currently happen.

Since the trees are many tens of megabytes, you should prefer to start from something already at hand. If you have a -RELEASE CD, you can copy or extract an initial source from it. This will save a significant transfer of data.

You can recognize these „starter” deltas by the `x` appended to the number (`src-cur.3210XEmpty.gz` for instance). The designation following the `x` corresponds to the origin of your initial „seed”. `Empty` is an empty directory. As a rule a base transition from `Empty` is produced every 100 deltas. By the way, they are large! 70 to 80 Megabytes of `gzip`’d data is common for the `XEmpty` deltas.

Once you have picked a base delta to start from, you will also need all deltas with higher numbers following it.

A.4.4. Using CTM in Your Daily Life

To apply the deltas, simply say:

```
# cd /where/ever/you/want/the/stuff
# ctm -v -v /where/you/store/your/deltas/src-xxx.*
```

CTM understands deltas which have been put through `gzip`, so you do not need to `gunzip` them first, this saves disk space.

Unless it feels very secure about the entire process, **CTM** will not touch your tree. To verify a delta you can also use the `-c` flag and **CTM** will not actually touch your tree; it will merely verify the integrity of the delta and see if it would apply cleanly to your current tree.

There are other options to **CTM** as well, see the manual pages or look in the sources for more information.

That is really all there is to it. Every time you get a new delta, just run it through **CTM** to keep your sources up to date.

Do not remove the deltas if they are hard to download again. You just might want to keep them around in case something bad happens. Even if you only have floppy disks, consider using `fdwrite` to make a copy.

A.4.5. Keeping Your Local Changes

As a developer one would like to experiment with and change files in the source tree. **CTM** supports local modifications in a limited way: before checking for the presence of a file `foo`, it first looks for `foo.ctm`. If this file exists, **CTM** will operate on it instead of `foo`.

This behavior gives us a simple way to maintain local changes: simply copy the files you plan to modify to the corresponding file names with a `.ctm` suffix. Then you can freely hack the code, while **CTM** keeps the `.ctm` file up-to-date.

A.4.6. Other Interesting CTM Options

A.4.6.1. Finding Out Exactly What Would Be Touched by an Update

You can determine the list of changes that **CTM** will make on your source repository using the `-l` option to **CTM**.

This is useful if you would like to keep logs of the changes, pre- or post- process the modified files in any manner, or just are feeling a tad paranoid.

A.4.6.2. Making Backups Before Updating

Sometimes you may want to backup all the files that would be changed by a **CTM** update.

Specifying the `-B backup-file` option causes **CTM** to backup all files that would be touched by a given **CTM** delta to `backup-file`.

A.4.6.3. Restricting the Files Touched by an Update

Sometimes you would be interested in restricting the scope of a given **CTM** update, or may be interested in extracting just a few files from a sequence of deltas.

You can control the list of files that **CTM** would operate on by specifying filtering regular expressions using the `-e` and `-x` options.

For example, to extract an up-to-date copy of `lib/libc/Makefile` from your collection of saved **CTM** deltas, run the commands:

```
# cd /where/ever/you/want/to/extract/it/  
# ctm -e '^lib/libc/Makefile' ~ctm/src-xxx.*
```

For every file specified in a **CTM** delta, the `-e` and `-x` options are applied in the order given on the command line. The file is processed by **CTM** only if it is marked as eligible after all the `-e` and `-x` options are applied to it.

A.4.7. Future Plans for CTM

Tons of them:

- Use some kind of authentication into the **CTM** system, so as to allow detection of spoofed **CTM** updates.
- Clean up the options to **CTM**, they became confusing and counter intuitive.

A.4.8. Miscellaneous Stuff

There is a sequence of deltas for the `ports` collection too, but interest has not been all that high yet.

A.4.9. CTM Mirrors

CTM/FreeBSD is available via anonymous FTP from the following mirror sites. If you choose to obtain **CTM** via anonymous FTP, please try to use a site near you.

In case of problems, please contact the `ctm-users` (<http://lists.FreeBSD.org/mailman/listinfo/ctm-users>) mailing list.

California, Bay Area, official source

- <ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CTM/>

South Africa, backup server for old deltas

- <ftp://ftp.za.FreeBSD.org/pub/FreeBSD/CTM/>

Taiwan/R.O.C.

- <ftp://ctm.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>
- <ftp://ctm2.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>
- <ftp://ctm3.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>

If you did not find a mirror near to you or the mirror is incomplete, try to use a search engine such as alltheweb (<http://www.alltheweb.com/>).

A.5. Using CVSup

A.5.1. Introduction

CVSup is a software package for distributing and updating source trees from a master CVS repository on a remote server host. The FreeBSD sources are maintained in a CVS repository on a central development machine in California. With **CVSup**, FreeBSD users can easily keep their own source trees up to date.

CVSup uses the so-called *pull* model of updating. Under the pull model, each client asks the server for updates, if and when they are wanted. The server waits passively for update requests from its clients. Thus all updates are instigated by the client. The server never sends unsolicited updates. Users must either run the **CVSup** client manually to get an update, or they must set up a `cron` job to run it automatically on a regular basis.

The term **CVSup**, capitalized just so, refers to the entire software package. Its main components are the client `cvsup` which runs on each user's machine, and the server `cvsupd` which runs at each of the FreeBSD mirror sites.

As you read the FreeBSD documentation and mailing lists, you may see references to **sup**. **Sup** was the predecessor of **CVSup**, and it served a similar purpose. **CVSup** is used much in the same way as `sup` and, in fact, uses configuration files which are backward-compatible with `sup`'s. **Sup** is no longer used in the FreeBSD project, because **CVSup** is both faster and more flexible.

A.5.2. Installation

The easiest way to install **CVSup** is to use the precompiled `net/cvsup` package from the FreeBSD packages collection. If you prefer to build **CVSup** from source, you can use the `net/cvsup` port instead. But be forewarned: the `net/cvsup` port depends on the Modula-3 system, which takes a substantial amount of time and disk space to download and build.

Notatka: If you are going to be using **CVSup** on a machine which will not have **XFree86** or **Xorg** installed, such as a server, be sure to use the port which does not include the **CVSup** GUI, `net/cvsup-without-gui`.

A.5.3. CVSup Configuration

CVSup's operation is controlled by a configuration file called the `supfile`. There are some sample `supfiles` in the directory `/usr/share/examples/cvsup/`.

The information in a `supfile` answers the following questions for **CVSup**:

- Which files do you want to receive?
- Which versions of them do you want?
- Where do you want to get them from?

- Where do you want to put them on your own machine?
- Where do you want to put your status files?

In the following sections, we will construct a typical `supfile` by answering each of these questions in turn. First, we describe the overall structure of a `supfile`.

A `supfile` is a text file. Comments begin with `#` and extend to the end of the line. Lines that are blank and lines that contain only comments are ignored.

Each remaining line describes a set of files that the user wishes to receive. The line begins with the name of a „collection”, a logical grouping of files defined by the server. The name of the collection tells the server which files you want. After the collection name come zero or more fields, separated by white space. These fields answer the questions listed above. There are two types of fields: flag fields and value fields. A flag field consists of a keyword standing alone, e.g., `delete` or `compress`. A value field also begins with a keyword, but the keyword is followed without intervening white space by `=` and a second word. For example, `release=cvs` is a value field.

A `supfile` typically specifies more than one collection to receive. One way to structure a `supfile` is to specify all of the relevant fields explicitly for each collection. However, that tends to make the `supfile` lines quite long, and it is inconvenient because most fields are the same for all of the collections in a `supfile`. **CVSup** provides a defaulting mechanism to avoid these problems. Lines beginning with the special pseudo-collection name `*default` can be used to set flags and values which will be used as defaults for the subsequent collections in the `supfile`. A default value can be overridden for an individual collection, by specifying a different value with the collection itself. Defaults can also be changed or augmented in mid-`supfile` by additional `*default` lines.

With this background, we will now proceed to construct a `supfile` for receiving and updating the main source tree of FreeBSD-CURRENT.

- Which files do you want to receive?

The files available via **CVSup** are organized into named groups called „collections”. The collections that are available are described in the following section. In this example, we wish to receive the entire main source tree for the FreeBSD system. There is a single large collection `src-all` which will give us all of that. As a first step toward constructing our `supfile`, we simply list the collections, one per line (in this case, only one line):

```
src-all
```

- Which version(s) of them do you want?

With **CVSup**, you can receive virtually any version of the sources that ever existed. That is possible because the **cvsupd** server works directly from the CVS repository, which contains all of the versions. You specify which one of them you want using the `tag=` and `date=` value fields.

Ostrzeżenie Be very careful to specify any `tag=` fields correctly. Some tags are valid only for certain collections of files. If you specify an incorrect or misspelled tag, **CVSup** will delete files which you probably do not want deleted. In particular, use *only* `tag=.` for the `ports-*` collections.

The `tag=` field names a symbolic tag in the repository. There are two kinds of tags, revision tags and branch tags. A revision tag refers to a specific revision. Its meaning stays the same from day to day. A branch tag, on the other hand, refers to the latest revision on a given line of development, at any given time. Because a branch tag does not refer to a specific revision, it may mean something different tomorrow than it means today.

Sekcja A.7 contains branch tags that users might be interested in. When specifying a tag in **CVSup**'s configuration file, it must be preceded with `tag=` (`RELENG_4` will become `tag=RELENG_4`). Keep in mind that only the `tag=` is relevant for the Ports Collection.

Ostrzeżenie Be very careful to type the tag name exactly as shown. **CVSup** cannot distinguish between valid and invalid tags. If you misspell the tag, **CVSup** will behave as though you had specified a valid tag which happens to refer to no files at all. It will delete your existing sources in that case.

When you specify a branch tag, you normally receive the latest versions of the files on that line of development. If you wish to receive some past version, you can do so by specifying a date with the `date=` value field. The `cvsup(1)` manual page explains how to do that.

For our example, we wish to receive FreeBSD-CURRENT. We add this line at the beginning of our `supfile`:

```
*default tag=.
```

There is an important special case that comes into play if you specify neither a `tag=` field nor a `date=` field. In that case, you receive the actual RCS files directly from the server's CVS repository, rather than receiving a particular version. Developers generally prefer this mode of operation. By maintaining a copy of the repository itself on their systems, they gain the ability to browse the revision histories and examine past versions of files. This gain is achieved at a large cost in terms of disk space, however.

- Where do you want to get them from?

We use the `host=` field to tell `cvsup` where to obtain its updates. Any of the **CVSup** mirror sites will do, though you should try to select one that is close to you in cyberspace. In this example we will use a fictional FreeBSD distribution site, `cvsup99.FreeBSD.org`:

```
*default host=cvsup99.FreeBSD.org
```

You will need to change the host to one that actually exists before running **CVSup**. On any particular run of `cvsup`, you can override the host setting on the command line, with `-h hostname`.

- Where do you want to put them on your own machine?

The `prefix=` field tells `cvsup` where to put the files it receives. In this example, we will put the source files directly into our main source tree, `/usr/src`. The `src` directory is already implicit in the collections we have chosen to receive, so this is the correct specification:

```
*default prefix=/usr
```

- Where should `cvsup` maintain its status files?

The **CVSup** client maintains certain status files in what is called the „base” directory. These files help **CVSup** to work more efficiently, by keeping track of which updates you have already received. We will use the standard base directory, `/var/db`:

```
*default base=/var/db
```

If your base directory does not already exist, now would be a good time to create it. The `cvsup` client will refuse to run if the base directory does not exist.

- Miscellaneous `supfile` settings:

There is one more line of boiler plate that normally needs to be present in the `supfile`:

```
*default release=cvs delete use-rel-suffix compress
```

`release=cvs` indicates that the server should get its information out of the main FreeBSD CVS repository. This is virtually always the case, but there are other possibilities which are beyond the scope of this discussion.

`delete` gives **CVSup** permission to delete files. You should always specify this, so that **CVSup** can keep your source tree fully up-to-date. **CVSup** is careful to delete only those files for which it is responsible. Any extra files you happen to have will be left strictly alone.

`use-rel-suffix` is ... arcane. If you really want to know about it, see the `cvsup(1)` manual page. Otherwise, just specify it and do not worry about it.

`compress` enables the use of `gzip`-style compression on the communication channel. If your network link is T1 speed or faster, you probably should not use compression. Otherwise, it helps substantially.

- Putting it all together:

Here is the entire `supfile` for our example:

```
*default tag=.
*default host=cvsup99.FreeBSD.org
*default prefix=/usr
*default base=/var/db
*default release=cvs delete use-rel-suffix compress

src-all
```

A.5.3.1. The `refuse` File

As mentioned above, **CVSup** uses a *pull method*. Basically, this means that you connect to the **CVSup** server, and it says, „Here is what you can download from me...”, and your client responds „OK, I will take this, this, this, and this.” In the default configuration, the **CVSup** client will take every file associated with the collection and tag you chose in the configuration file. However, this is not always what you want, especially if you are synching the `doc`, `ports`, or `www` trees — most people cannot read four or five languages, and therefore they do not need to download the language-specific files. If you are **CVSup**ing the Ports Collection, you can get around this by specifying each collection individually (e.g., *ports-astrology*, *ports-biology*, etc instead of simply saying *ports-all*). However, since the `doc` and `www` trees do not have language-specific collections, you must use one of **CVSup**’s many nifty features: the `refuse` file.

The `refuse` file essentially tells **CVSup** that it should not take every single file from a collection; in other words, it tells the client to *refuse* certain files from the server. The `refuse` file can be found (or, if you do not yet have one, should be placed) in `base/sup/`. `base` is defined in your `supfile`; our defined `base` is `/var/db`, which means that by default the `refuse` file is `/var/db/sup/refuse`.

The `refuse` file has a very simple format; it simply contains the names of files or directories that you do not wish to download. For example, if you cannot speak any languages other than English and some German, and you do not feel the need to read the German translation of documentation, you can put the following in your `refuse` file:

```
doc/bn_*
doc/da_*
doc/de_*
doc/el_*
doc/es_*
doc/fr_*
doc/it_*
doc/ja_*
doc/nl_*
```

```
doc/no_*
doc/pl_*
doc/pt_*
doc/ru_*
doc/sr_*
doc/tr_*
doc/zh_*
```

and so forth for the other languages (you can find the full list by browsing the FreeBSD CVS repository (<http://www.FreeBSD.org/cgi/cvsweb.cgi/>)).

With this very useful feature, those users who are on slow links or pay by the minute for their Internet connection will be able to save valuable time as they will no longer need to download files that they will never use. For more information on `refuse` files and other neat features of **CVSup**, please view its manual page.

A.5.4. Running CVSup

You are now ready to try an update. The command line for doing this is quite simple:

```
# cvsup supfile
```

where *supfile* is of course the name of the *supfile* you have just created. Assuming you are running under X11, **cvsup** will display a GUI window with some buttons to do the usual things. Press the **go** button, and watch it run.

Since you are updating your actual `/usr/src` tree in this example, you will need to run the program as `root` so that **cvsup** has the permissions it needs to update your files. Having just created your configuration file, and having never used this program before, that might understandably make you nervous. There is an easy way to do a trial run without touching your precious files. Just create an empty directory somewhere convenient, and name it as an extra argument on the command line:

```
# mkdir /var/tmp/dest
# cvsup supfile /var/tmp/dest
```

The directory you specify will be used as the destination directory for all file updates. **CVSup** will examine your usual files in `/usr/src`, but it will not modify or delete any of them. Any file updates will instead land in `/var/tmp/dest/usr/src`. **CVSup** will also leave its base directory status files untouched when run this way. The new versions of those files will be written into the specified directory. As long as you have read access to `/usr/src`, you do not even need to be `root` to perform this kind of trial run.

If you are not running X11 or if you just do not like GUIs, you should add a couple of options to the command line when you run **cvsup**:

```
# cvsup -g -L 2 supfile
```

The `-g` tells **CVSup** not to use its GUI. This is automatic if you are not running X11, but otherwise you have to specify it.

The `-L 2` tells **CVSup** to print out the details of all the file updates it is doing. There are three levels of verbosity, from `-L 0` to `-L 2`. The default is 0, which means total silence except for error messages.

There are plenty of other options available. For a brief list of them, type `cvsup -h`. For more detailed descriptions, see the manual page.

Once you are satisfied with the way updates are working, you can arrange for regular runs of **CVSup** using `cron(8)`. Obviously, you should not let **CVSup** use its GUI when running it from `cron(8)`.

A.5.5. CVSup File Collections

The file collections available via **CVSup** are organized hierarchically. There are a few large collections, and they are divided into smaller sub-collections. Receiving a large collection is equivalent to receiving each of its sub-collections. The hierarchical relationships among collections are reflected by the use of indentation in the list below.

The most commonly used collections are `src-all`, and `ports-all`. The other collections are used only by small groups of people for specialized purposes, and some mirror sites may not carry all of them.

```
cvs-all release=cvs
```

The main FreeBSD CVS repository, including the cryptography code.

```
distrib release=cvs
```

Files related to the distribution and mirroring of FreeBSD.

```
doc-all release=cvs
```

Sources for the FreeBSD Handbook and other documentation. This does not include files for the FreeBSD web site.

```
ports-all release=cvs
```

The FreeBSD Ports Collection.

WAŻNE: If you do not want to update the whole of `ports-all` (the whole ports tree), but use one of the subcollections listed below, make sure that you *always* update the `ports-base` subcollection! Whenever something changes in the ports build infrastructure represented by `ports-base`, it is virtually certain that those changes will be used by „real” ports real soon. Thus, if you only update the „real” ports and they use some of the new features, there is a very high chance that their build will fail with some mysterious error message. The *very first* thing to do in this case is to make sure that your `ports-base` subcollection is up to date.

WAŻNE: If you are going to be building your own local copy of `ports/INDEX`, you *must* accept `ports-all` (the whole ports tree). Building `ports/INDEX` with a partial tree is not supported. See the FAQ (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/books/faq/applications.html#MAKE-INDEX).

```
ports-accessibility release=cvs
```

Software to help disabled users.

```
ports-arabic release=cvs
```

Arabic language support.

`ports-archivers release=cvs`

Archiving tools.

`ports-astro release=cvs`

Astronomical ports.

`ports-audio release=cvs`

Sound support.

`ports-base release=cvs`

The Ports Collection build infrastructure - various files located in the `Mk/` and `Tools/` subdirectories of `/usr/ports`.

Notatka: Please see the important warning above: you should *a/ways* update this subcollection, whenever you update any part of the FreeBSD Ports Collection!

`ports-benchmarks release=cvs`

Benchmarks.

`ports-biology release=cvs`

Biology.

`ports-cad release=cvs`

Computer aided design tools.

`ports-chinese release=cvs`

Chinese language support.

`ports-comms release=cvs`

Communication software.

`ports-converters release=cvs`

character code converters.

`ports-databases release=cvs`

Databases.

`ports-deskutils release=cvs`

Things that used to be on the desktop before computers were invented.

`ports-devel release=cvs`

Development utilities.

`ports-dns release=cvs`

DNS related software.

`ports-editors release=cvs`

Editors.

`ports-emulators release=cvs`

Emulators for other operating systems.

`ports-finance release=cvs`

Monetary, financial and related applications.

`ports-ftp release=cvs`

FTP client and server utilities.

`ports-games release=cvs`

Games.

`ports-german release=cvs`

German language support.

`ports-graphics release=cvs`

Graphics utilities.

`ports-hebrew release=cvs`

Hebrew language support.

`ports-hungarian release=cvs`

Hungarian language support.

`ports-irc release=cvs`

Internet Relay Chat utilities.

`ports-japanese release=cvs`

Japanese language support.

`ports-java release=cvs`

Java utilities.

`ports-korean release=cvs`

Korean language support.

`ports-lang release=cvs`

Programming languages.

ports-mail release=cvs

Mail software.

ports-math release=cvs

Numerical computation software.

ports-mbone release=cvs

MBone applications.

ports-misc release=cvs

Miscellaneous utilities.

ports-multimedia release=cvs

Multimedia software.

ports-net release=cvs

Networking software.

ports-net-im release=cvs

Instant messaging software.

ports-net-mgmt release=cvs

Network management software.

ports-net-p2p release=cvs

Peer to peer networking.

ports-news release=cvs

USENET news software.

ports-palm release=cvs

Software support for Palm™ series.

ports-polish release=cvs

Polish language support.

ports-portuguese release=cvs

Portuguese language support.

ports-print release=cvs

Printing software.

ports-russian release=cvs

Russian language support.

ports-science release=cvsv

Science.

ports-security release=cvsv

Security utilities.

ports-shells release=cvsv

Command line shells.

ports-sysutils release=cvsv

System utilities.

ports-textproc release=cvsv

text processing utilities (does not include desktop publishing).

ports-ukrainian release=cvsv

Ukrainian language support.

ports-vietnamese release=cvsv

Vietnamese language support.

ports-www release=cvsv

Software related to the World Wide Web.

ports-x11 release=cvsv

Ports to support the X window system.

ports-x11-clocks release=cvsv

X11 clocks.

ports-x11-fm release=cvsv

X11 file managers.

ports-x11-fonts release=cvsv

X11 fonts and font utilities.

ports-x11-toolkits release=cvsv

X11 toolkits.

ports-x11-servers release=cvsv

X11 servers.

ports-x11-themes release=cvsv

X11 themes.

`ports-x11-wm release=cvs`

X11 window managers.

`projects-all release=cvs`

Sources for the FreeBSD projects repository.

`src-all release=cvs`

The main FreeBSD sources, including the cryptography code.

`src-base release=cvs`

Miscellaneous files at the top of `/usr/src`.

`src-bin release=cvs`

User utilities that may be needed in single-user mode (`/usr/src/bin`).

`src-contrib release=cvs`

Utilities and libraries from outside the FreeBSD project, used relatively unmodified (`/usr/src/contrib`).

`src-crypto release=cvs`

Cryptography utilities and libraries from outside the FreeBSD project, used relatively unmodified (`/usr/src/crypto`).

`src-eBones release=cvs`

Kerberos and DES (`/usr/src/eBones`). Not used in current releases of FreeBSD.

`src-etc release=cvs`

System configuration files (`/usr/src/etc`).

`src-games release=cvs`

Games (`/usr/src/games`).

`src-gnu release=cvs`

Utilities covered by the GNU Public License (`/usr/src/gnu`).

`src-include release=cvs`

Header files (`/usr/src/include`).

`src-kerberos5 release=cvs`

Kerberos5 security package (`/usr/src/kerberos5`).

`src-kerberosIV release=cvs`

KerberosIV security package (`/usr/src/kerberosIV`).

`src-lib release=cvs`

Libraries (`/usr/src/lib`).

`src-libexec release=cvs`

System programs normally executed by other programs (`/usr/src/libexec`).

`src-release release=cvs`

Files required to produce a FreeBSD release (`/usr/src/release`).

`src-sbin release=cvs`

System utilities for single-user mode (`/usr/src/sbin`).

`src-secure release=cvs`

Cryptographic libraries and commands (`/usr/src/secure`).

`src-share release=cvs`

Files that can be shared across multiple systems (`/usr/src/share`).

`src-sys release=cvs`

The kernel (`/usr/src/sys`).

`src-sys-crypto release=cvs`

Kernel cryptography code (`/usr/src/sys/crypto`).

`src-tools release=cvs`

Various tools for the maintenance of FreeBSD (`/usr/src/tools`).

`src-usrbin release=cvs`

User utilities (`/usr/src/usr.bin`).

`src-usrsbin release=cvs`

System utilities (`/usr/src/usr.sbin`).

`www release=cvs`

The sources for the FreeBSD WWW site.

`distrib release=self`

The **CVSup** server's own configuration files. Used by **CVSup** mirror sites.

`gnats release=current`

The GNATS bug-tracking database.

`mail-archive release=current`

FreeBSD mailing list archive.

`www release=current`

The pre-processed FreeBSD WWW site files (not the source files). Used by WWW mirror sites.

A.5.6. For More Information

For the **CVSup** FAQ and other information about **CVSup**, see The CVSup Home Page (<http://www.polstra.com/projects/freeware/CVSup/>).

Most FreeBSD-related discussion of **CVSup** takes place on the Techniczna lista dyskusyjna FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>). New versions of the software are announced there, as well as on the Informacyjna lista dyskusyjna FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce>).

Questions and bug reports should be addressed to the author of the program at [<cvsup-bugs@polstra.com>](mailto:cvsup-bugs@polstra.com).

A.5.7. CVSup Sites

CVSup servers for FreeBSD are running at the following sites:

Central Servers, Primary Mirror Sites, Armenia, Australia, Austria, Brazil, Canada, China, Costa Rica, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Israel, Italy, Japan, Korea, Kuwait, Kyrgyzstan, Latvia, Lithuania, Netherlands, New Zealand, Norway, Philippines, Poland, Portugal, Romania, Russia, San Marino, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Thailand, Turkey, Ukraine, United Kingdom, USA.

(as of UTC)

Central Servers

- cvsup.FreeBSD.org

Primary Mirror Sites

- cvsup1.FreeBSD.org
- cvsup2.FreeBSD.org
- cvsup3.FreeBSD.org
- cvsup4.FreeBSD.org
- cvsup5.FreeBSD.org
- cvsup6.FreeBSD.org
- cvsup7.FreeBSD.org
- cvsup8.FreeBSD.org

- cvsup9.FreeBSD.org
- cvsup10.FreeBSD.org
- cvsup11.FreeBSD.org
- cvsup12.FreeBSD.org
- cvsup13.FreeBSD.org
- cvsup14.FreeBSD.org
- cvsup15.FreeBSD.org
- cvsup16.FreeBSD.org
- cvsup18.FreeBSD.org

Armenia

- cvsup1.am.FreeBSD.org

Australia

- cvsup.au.FreeBSD.org

Austria

- cvsup.at.FreeBSD.org

Brazil

- cvsup.br.FreeBSD.org
- cvsup2.br.FreeBSD.org
- cvsup3.br.FreeBSD.org
- cvsup4.br.FreeBSD.org
- cvsup5.br.FreeBSD.org

Canada

- cvsup1.ca.FreeBSD.org

China

- cvsup.cn.FreeBSD.org
- cvsup2.cn.FreeBSD.org

Costa Rica

- cvsup1.cr.FreeBSD.org

Czech Republic

- cvsup.cz.FreeBSD.org

Denmark

- cvsup.dk.FreeBSD.org
- cvsup2.dk.FreeBSD.org

Estonia

- cvsup.ee.FreeBSD.org

Finland

- cvsup.fi.FreeBSD.org
- cvsup2.fi.FreeBSD.org

France

- cvsup.fr.FreeBSD.org
- cvsup1.fr.FreeBSD.org
- cvsup2.fr.FreeBSD.org

- cvsup3.fr.FreeBSD.org
- cvsup4.fr.FreeBSD.org
- cvsup5.fr.FreeBSD.org
- cvsup8.fr.FreeBSD.org

Germany

- cvsup.de.FreeBSD.org
- cvsup2.de.FreeBSD.org
- cvsup3.de.FreeBSD.org
- cvsup4.de.FreeBSD.org
- cvsup5.de.FreeBSD.org
- cvsup6.de.FreeBSD.org
- cvsup7.de.FreeBSD.org
- cvsup8.de.FreeBSD.org

Greece

- cvsup.gr.FreeBSD.org
- cvsup2.gr.FreeBSD.org

Hungary

- cvsup.hu.FreeBSD.org

Iceland

- cvsup.is.FreeBSD.org

Ireland

- cvsup.ie.FreeBSD.org
- cvsup2.ie.FreeBSD.org

Israel

- cvsup.il.FreeBSD.org

Italy

- cvsup.it.FreeBSD.org

Japan

- cvsup.jp.FreeBSD.org
- cvsup2.jp.FreeBSD.org
- cvsup3.jp.FreeBSD.org
- cvsup4.jp.FreeBSD.org
- cvsup5.jp.FreeBSD.org
- cvsup6.jp.FreeBSD.org

Korea

- cvsup.kr.FreeBSD.org
- cvsup2.kr.FreeBSD.org
- cvsup3.kr.FreeBSD.org

Kuwait

- cvsup1.kw.FreeBSD.org

Kyrgyzstan

- cvsup.kg.FreeBSD.org

Latvia

- cvsup.lv.FreeBSD.org
- cvsup2.lv.FreeBSD.org

Lithuania

- cvsup.lt.FreeBSD.org
- cvsup2.lt.FreeBSD.org
- cvsup3.lt.FreeBSD.org

Netherlands

- cvsup.nl.FreeBSD.org
- cvsup2.nl.FreeBSD.org
- cvsup3.nl.FreeBSD.org

New Zealand

- cvsup.nz.FreeBSD.org

Norway

- cvsup.no.FreeBSD.org

Philippines

- cvsup1.ph.FreeBSD.org

Poland

- cvsup.pl.FreeBSD.org
- cvsup2.pl.FreeBSD.org

- cvsup3.pl.FreeBSD.org

Portugal

- cvsup.pt.FreeBSD.org
- cvsup2.pt.FreeBSD.org
- cvsup3.pt.FreeBSD.org

Romania

- cvsup.ro.FreeBSD.org
- cvsup1.ro.FreeBSD.org
- cvsup2.ro.FreeBSD.org
- cvsup3.ro.FreeBSD.org

Russia

- cvsup.ru.FreeBSD.org
- cvsup2.ru.FreeBSD.org
- cvsup3.ru.FreeBSD.org
- cvsup4.ru.FreeBSD.org
- cvsup5.ru.FreeBSD.org
- cvsup6.ru.FreeBSD.org
- cvsup7.ru.FreeBSD.org

San Marino

- cvsup.sm.FreeBSD.org

Slovak Republic

- cvsup.sk.FreeBSD.org

Slovenia

- cvsup.si.FreeBSD.org
- cvsup2.si.FreeBSD.org

South Africa

- cvsup.za.FreeBSD.org
- cvsup2.za.FreeBSD.org

Spain

- cvsup.es.FreeBSD.org
- cvsup2.es.FreeBSD.org
- cvsup3.es.FreeBSD.org

Sweden

- cvsup.se.FreeBSD.org
- cvsup2.se.FreeBSD.org

Switzerland

- cvsup.ch.FreeBSD.org

Taiwan

- cvsup.tw.FreeBSD.org
- cvsup3.tw.FreeBSD.org
- cvsup4.tw.FreeBSD.org
- cvsup5.tw.FreeBSD.org
- cvsup6.tw.FreeBSD.org
- cvsup7.tw.FreeBSD.org

- cvsup8.tw.FreeBSD.org
- cvsup9.tw.FreeBSD.org
- cvsup10.tw.FreeBSD.org
- cvsup11.tw.FreeBSD.org
- cvsup12.tw.FreeBSD.org
- cvsup13.tw.FreeBSD.org
- cvsup14.tw.FreeBSD.org

Thailand

- cvsup.th.FreeBSD.org

Turkey

- cvsup.tr.FreeBSD.org
- cvsup2.tr.FreeBSD.org

Ukraine

- cvsup2.ua.FreeBSD.org
- cvsup3.ua.FreeBSD.org
- cvsup5.ua.FreeBSD.org
- cvsup6.ua.FreeBSD.org
- cvsup7.ua.FreeBSD.org

United Kingdom

- cvsup.uk.FreeBSD.org
- cvsup2.uk.FreeBSD.org
- cvsup3.uk.FreeBSD.org
- cvsup4.uk.FreeBSD.org

USA

- cvsup1.us.FreeBSD.org
- cvsup2.us.FreeBSD.org
- cvsup3.us.FreeBSD.org
- cvsup4.us.FreeBSD.org
- cvsup5.us.FreeBSD.org
- cvsup6.us.FreeBSD.org
- cvsup7.us.FreeBSD.org
- cvsup8.us.FreeBSD.org
- cvsup9.us.FreeBSD.org
- cvsup10.us.FreeBSD.org
- cvsup11.us.FreeBSD.org
- cvsup12.us.FreeBSD.org
- cvsup13.us.FreeBSD.org
- cvsup14.us.FreeBSD.org
- cvsup15.us.FreeBSD.org
- cvsup16.us.FreeBSD.org
- cvsup18.us.FreeBSD.org

A.6. Using Portsnap

A.6.1. Introduction

Portsnap is a system for securely distributing the FreeBSD ports tree. Approximately once an hour, a „snapshot” of the ports tree is generated, repackaged, and cryptographically signed. The resulting files are then distributed via HTTP.

Like **CVSup**, **Portsnap** uses a *pull* model of updating: The packaged and signed ports trees are placed on a web server which waits passively for clients to request files. Users must either run `portsnap(8)` manually to download updates or set up a `cron(8)` job to download updates automatically on a regular basis.

For technical reasons, **Portsnap** does not update the „live” ports tree in `/usr/ports/` directly; instead, it works via a compressed copy of the ports tree stored in `/var/db/portsnap/` by default. This compressed copy is then used to update the live ports tree.

Notatka: If **Portsnap** is installed from the FreeBSD Ports Collection, then the default location for its compressed snapshot will be `/usr/local/portsnap/` instead of `/var/db/portsnap/`.

A.6.2. Installation

On FreeBSD 6.0 and more recent versions, **Portsnap** is contained in the FreeBSD base system. On older versions of FreeBSD, it can be installed using the `sysutils/portsnap` port.

A.6.3. Portsnap Configuration

Portsnap's operation is controlled by the `/etc/portsnap.conf` configuration file. For most users, the default configuration file will suffice; for more details, consult the `portsnap.conf(5)` manual page.

Notatka: If **Portsnap** is installed from the FreeBSD Ports Collection, it will use the configuration file `/usr/local/etc/portsnap.conf` instead of `/etc/portsnap.conf`. This configuration file is not created when the port is installed, but a sample configuration file is distributed; to copy it into place, run the following command:

```
# cd /usr/local/etc && cp portsnap.conf.sample portsnap.conf
```

A.6.4. Running Portsnap for the First Time

The first time `portsnap(8)` is run, it will need to download a compressed snapshot of the entire ports tree into `/var/db/portsnap/` (or `/usr/local/portsnap/` if **Portsnap** was installed from the Ports Collection). For the beginning of 2006 this is approximately a 41 MB download.

```
# portsnap fetch
```

Once the compressed snapshot has been downloaded, a „live” copy of the ports tree can be extracted into `/usr/ports/`. This is necessary even if a ports tree has already been created in that directory (e.g., by using **CVSup**), since it establishes a baseline from which `portsnap` can determine which parts of the ports tree need to be updated later.

```
# portsnap extract
```

Notatka: In the default installation `/usr/ports` is not created. If you run FreeBSD 6.0-RELEASE, it should be created before `portsnap` is used. On more recent versions of FreeBSD or **Portsnap**, this operation will be done automatically at first use of the `portsnap` command.

A.6.5. Updating the Ports Tree

After an initial compressed snapshot of the ports tree has been downloaded and extracted into `/usr/ports/`, updating the ports tree consists of two steps: *fetching* updates to the compressed snapshot, and using them to *update* the live ports tree. These two steps can be specified to `portsnap` as a single command:

```
# portsnap fetch update
```

Notatka: Some older versions of `portsnap` do not support this syntax; if it fails, try instead the following:

```
# portsnap fetch
# portsnap update
```

A.6.6. Running Portsnap from cron

In order to avoid problems with „flash crowds” accessing the **Portsnap** servers, `portsnap fetch` will not run from a `cron(8)` job. Instead, a special `portsnap cron` command exists, which waits for a random duration up to 3600 seconds before fetching updates.

In addition, it is strongly recommended that `portsnap update` not be run from a `cron` job, since it is liable to cause major problems if it happens to run at the same time as a port is being built or installed. However, it is safe to update the ports’ `INDEX` files, and this can be done by passing the `-I` flag to `portsnap`. (Obviously, if `portsnap -I update` is run from `cron`, then it will be necessary to run `portsnap update` without the `-I` flag at a later time in order to update the rest of the tree.)

Adding the following line to `/etc/crontab` will cause `portsnap` to update its compressed snapshot and the `INDEX` files in `/usr/ports/`, and will send an email if any installed ports are out of date:

```
0 3 * * * root portsnap -I cron update && pkg_version -vIL=
```

Notatka: If the system clock is not set to the local time zone, please replace `3` with a random value between `0` and `23`, in order to spread the load on the **Portsnap** servers more evenly.

Notatka: Some older versions of `portsnap` do not support listing multiple commands (e.g., `cron update`) in the same invocation of `portsnap`. If the line above fails, try replacing `portsnap -I cron update` with `portsnap cron && portsnap -I update`.

A.7. CVS Tags

When obtaining or updating sources using `cv`s or **CVSup**, a revision tag must be specified. A revision tag refers to either a particular line of FreeBSD development, or a specific point in time. The first type are called „branch tags”, and the second type are called „release tags”.

A.7.1. Branch Tags

All of these, with the exception of `HEAD` (which is always a valid tag), only apply to the `src/` tree. The `ports/`, `doc/`, and `www/` trees are not branched.

HEAD

Symbolic name for the main line, or FreeBSD-CURRENT. Also the default when no revision is specified.

In **CVSup**, this tag is represented by a `.` (not punctuation, but a literal `.` character).

Notatka: In CVS, this is the default when no revision tag is specified. It is usually *not* a good idea to checkout or update to CURRENT sources on a STABLE machine, unless that is your intent.

RELENG_6

The line of development for FreeBSD-6.X, also known as FreeBSD 6-STABLE

RELENG_6_1

The release branch for FreeBSD-6.1, used only for security advisories and other critical fixes.

RELENG_6_0

The release branch for FreeBSD-6.0, used only for security advisories and other critical fixes.

RELENG_5

The line of development for FreeBSD-5.X, also known as FreeBSD 5-STABLE.

RELENG_5_5

The release branch for FreeBSD-5.5, used only for security advisories and other critical fixes.

RELENG_5_4

The release branch for FreeBSD-5.4, used only for security advisories and other critical fixes.

RELENG_5_3

The release branch for FreeBSD-5.3, used only for security advisories and other critical fixes.

RELENG_5_2

The release branch for FreeBSD-5.2 and FreeBSD-5.2.1, used only for security advisories and other critical fixes.

RELENG_5_1

The release branch for FreeBSD-5.1, used only for security advisories and other critical fixes.

RELENG_5_0

The release branch for FreeBSD-5.0, used only for security advisories and other critical fixes.

RELENG_4

The line of development for FreeBSD-4.X, also known as FreeBSD 4-STABLE.

RELENG_4_11

The release branch for FreeBSD-4.11, used only for security advisories and other critical fixes.

RELENG_4_10

The release branch for FreeBSD-4.10, used only for security advisories and other critical fixes.

RELENG_4_9

The release branch for FreeBSD-4.9, used only for security advisories and other critical fixes.

RELENG_4_8

The release branch for FreeBSD-4.8, used only for security advisories and other critical fixes.

RELENG_4_7

The release branch for FreeBSD-4.7, used only for security advisories and other critical fixes.

RELENG_4_6

The release branch for FreeBSD-4.6 and FreeBSD-4.6.2, used only for security advisories and other critical fixes.

RELENG_4_5

The release branch for FreeBSD-4.5, used only for security advisories and other critical fixes.

RELENG_4_4

The release branch for FreeBSD-4.4, used only for security advisories and other critical fixes.

RELENG_4_3

The release branch for FreeBSD-4.3, used only for security advisories and other critical fixes.

RELENG_3

The line of development for FreeBSD-3.X, also known as 3.X-STABLE.

RELENG_2_2

The line of development for FreeBSD-2.2.X, also known as 2.2-STABLE. This branch is mostly obsolete.

A.7.2. Release Tags

These tags refer to a specific point in time when a particular version of FreeBSD was released. The release engineering process is documented in more detail by the Release Engineering Information (<http://www.FreeBSD.org/releng/>) and Release Process (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/releng/release-proc.html) documents. The `src` tree uses tag names that start with `RELENG_` tags. The `ports` and `doc` trees use tags whose names begin with `RELEASE` tags. Finally, the `www` tree is not tagged with any special name for releases.

RELENG_6_1_0_RELEASE

FreeBSD 6.1

RELENG_6_0_0_RELEASE

FreeBSD 6.0

RELENG_5_5_0_RELEASE

FreeBSD 5.5

RELENG_5_4_0_RELEASE

FreeBSD 5.4

RELENG_4_11_0_RELEASE

FreeBSD 4.11

RELENG_5_3_0_RELEASE

FreeBSD 5.3

RELENG_4_10_0_RELEASE

FreeBSD 4.10

RELENG_5_2_1_RELEASE

FreeBSD 5.2.1

RELENG_5_2_0_RELEASE

FreeBSD 5.2

RELENG_4_9_0_RELEASE

FreeBSD 4.9

RELENG_5_1_0_RELEASE

FreeBSD 5.1

RELENG_4_8_0_RELEASE

FreeBSD 4.8

RELENG_5_0_0_RELEASE

FreeBSD 5.0

RELENG_4_7_0_RELEASE

FreeBSD 4.7

RELENG_4_6_2_RELEASE

FreeBSD 4.6.2

RELENG_4_6_1_RELEASE

FreeBSD 4.6.1

RELENG_4_6_0_RELEASE

FreeBSD 4.6

RELENG_4_5_0_RELEASE

FreeBSD 4.5

RELENG_4_4_0_RELEASE

FreeBSD 4.4

RELENG_4_3_0_RELEASE

FreeBSD 4.3

RELENG_4_2_0_RELEASE

FreeBSD 4.2

RELENG_4_1_1_RELEASE

FreeBSD 4.1.1

RELENG_4_1_0_RELEASE

FreeBSD 4.1

RELENG_4_0_0_RELEASE

FreeBSD 4.0

RELENG_3_5_0_RELEASE

FreeBSD-3.5

RELENG_3_4_0_RELEASE

FreeBSD-3.4

RELENG_3_3_0_RELEASE

FreeBSD-3.3

RELENG_3_2_0_RELEASE

FreeBSD-3.2

RELENG_3_1_0_RELEASE

FreeBSD-3.1

RELENG_3_0_0_RELEASE

FreeBSD-3.0

RELENG_2_2_8_RELEASE

FreeBSD-2.2.8

RELENG_2_2_7_RELEASE

FreeBSD-2.2.7

RELENG_2_2_6_RELEASE

FreeBSD-2.2.6

RELENG_2_2_5_RELEASE

FreeBSD-2.2.5

RELENG_2_2_2_RELEASE

FreeBSD-2.2.2

RELENG_2_2_1_RELEASE

FreeBSD-2.2.1

RELENG_2_2_0_RELEASE

FreeBSD-2.2.0

A.8. AFS Sites

AFS servers for FreeBSD are running at the following sites:

Sweden

The path to the files are: `/afs/stacken.kth.se/ftp/pub/FreeBSD/`

<code>stacken.kth.se</code>	<code># Stacken Computer Club, KTH, Sweden</code>
<code>130.237.234.43</code>	<code>#hot.stacken.kth.se</code>
<code>130.237.237.230</code>	<code>#fishburger.stacken.kth.se</code>
<code>130.237.234.3</code>	<code>#milko.stacken.kth.se</code>

Maintainer `<ftp@stacken.kth.se>`

A.9. rsync Sites

The following sites make FreeBSD available through the rsync protocol. The **rsync** utility works in much the same way as the `rcp(1)` command, but has more options and uses the rsync remote-update protocol which transfers only the differences between two sets of files, thus greatly speeding up the synchronization over the network. This is most useful if you are a mirror site for the FreeBSD FTP server, or the CVS repository. The **rsync** suite is available for many operating systems, on FreeBSD, see the `net/rsync` port or use the package.

Czech Republic

`rsync://ftp.cz.FreeBSD.org/`

Available collections:

- `ftp`: A partial mirror of the FreeBSD FTP server.

- FreeBSD: A full mirror of the FreeBSD FTP server.

Germany

`rsync://grappa.unix-ag.uni-kl.de/`

Available collections:

- `freebsd-cvs`: The full FreeBSD CVS repository.

This machine also mirrors the CVS repositories of the NetBSD and the OpenBSD projects, among others.

Netherlands

`rsync://ftp.nl.FreeBSD.org/`

Available collections:

- `vol/4/freebsd-core`: A full mirror of the FreeBSD FTP server.

United Kingdom

`rsync://rsync.mirror.ac.uk/`

Available collections:

- `ftp.FreeBSD.org`: A full mirror of the FreeBSD FTP server.

United States of America

`rsync://ftp-master.FreeBSD.org/`

This server may only be used by FreeBSD primary mirror sites.

Available collections:

- `FreeBSD`: The master archive of the FreeBSD FTP server.
- `acl`: The FreeBSD master ACL list.

`rsync://ftp13.FreeBSD.org/`

Available collections:

- `FreeBSD`: A full mirror of the FreeBSD FTP server.

Dodatek B. Bibliografia

Podczas gdy strony podręcznika systemowego dostarczają szczegółowych informacji odnośnie każdego z elementów systemu operacyjnego FreeBSD, bardzo często nie przedstawiają one jak złożyć wszystkie te elementy razem tak, by usprawnić pracę całego systemu. Z tego właśnie powodu nie istnieje żaden substytut dobrej książki o administracji systemami UNIX i dobrego podręcznika ich użytkowania.

B.1. Książki i czasopisma poświęcone FreeBSD

Międzynarodowe książki i czasopisma:

- Using FreeBSD (<http://jdli.tw.FreeBSD.org/publication/book/freebsd2/index.htm>) (po chińsku).
- FreeBSD Unleashed (chiński przekład), wydany przez China Machine Press (<http://www.hzbook.com/>). ISBN 7-111-10201-0.
- FreeBSD From Scratch First Edition (po chińsku), wydany przez China Machine Press. ISBN 7-111-07482-3.
- FreeBSD From Scratch Second Edition (po chińsku), wydany przez China Machine Press. ISBN 7-111-10286-X.
- FreeBSD Handbook (chiński przekład), wydany przez Posts & Telecom Press (<http://www.ptpress.com.cn/>). ISBN 7-115-10541-3.
- FreeBSD 3.x Internet (po chińsku), wydany przez Tsinghua University Press (<http://www.tup.tsinghua.edu.cn/>). ISBN 7-900625-66-6.
- FreeBSD & Windows (po chińsku), ISBN 7-113-03845-X
- FreeBSD Internet Services HOWTO (po chińsku), ISBN 7-113-03423-3
- FreeBSD for PC 98'ers (po japońsku), wydany przez SHUWA System Co, LTD. ISBN 4-87966-468-5 C3055 P2900E.
- FreeBSD (po japońsku), wydany przez CUTT. ISBN 4-906391-22-2 C3055 P2400E.
- Complete Introduction to FreeBSD (<http://www.shoeisha.com/book/Detail.asp?bid=650>) (po japońsku), wydany przez Shoeisha Co., Ltd (<http://www.shoeisha.co.jp/>). ISBN 4-88135-473-6 P3600E.
- Personal UNIX Starter Kit FreeBSD (<http://www.ascii.co.jp/pb/book1/shinkan/detail/1322785.html>) (po japońsku), wydany przez ASCII (<http://www.ascii.co.jp/>). ISBN 4-7561-1733-3 P3000E.
- FreeBSD Handbook (japoński przekład), wydany przez ASCII (<http://www.ascii.co.jp/>). ISBN 4-7561-1580-2 P3800E.
- FreeBSD mit Methode (po niemiecku), wydany przez Computer und Literatur Verlag (<http://www.cul.de/>)Vertrieb Hanser, 1998. ISBN 3-932311-31-0.
- FreeBSD 4 - Installieren, Konfigurieren, Administrieren (<http://www.cul.de/freebsd.html>) (po niemiecku), wydany przez Computer und Literatur Verlag (<http://www.cul.de/>), 2001. ISBN 3-932311-88-4.
- FreeBSD 5 - Installieren, Konfigurieren, Administrieren (<http://www.cul.de/freebsd.html>) (po niemiecku), wydany przez Computer und Literatur Verlag (<http://www.cul.de/>), 2003. ISBN 3-936546-06-1.
- FreeBSD de Luxe (<http://www.mitp.de/vmi/mitp/detail/pWert/1343/>) (po niemiecku), wydany przez Verlag Modere Industrie (<http://www.mitp.de/>), 2003. ISBN 3-8266-1343-0.

- FreeBSD Install and Utilization Manual (<http://www.pc.mycom.co.jp/FreeBSD/install-manual.html>) (po japońsku), wydany przez Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>).
- Onno W Purbo, Dodi Maryanto, Syahrial Hubbany, Widjil Widodo *Building Internet Server with FreeBSD* (<http://maxwell.itb.ac.id/>) (w języku indonezyjskim), wydany przez Elex Media Komputindo (<http://www.elexmedia.co.id/>).

Książki i czasopisma w języku angielskim:

- Absolute BSD: The Ultimate Guide to FreeBSD (<http://www.AbsoluteBSD.com/>), wydany przez No Starch Press (<http://www.nostarch.com/>), 2002. ISBN: 1886411743
- The Complete FreeBSD (<http://www.freebsdmail.com/cgi-bin/fm/bsdcomp>), wydany przez O'Reilly (<http://www.oreilly.com/>), 2003. ISBN: 0596005164
- The FreeBSD Corporate Networker's Guide (<http://www.freebsd-corp-net-guide.com/>), wydany przez Addison-Wesley (<http://www.awl.com/aw/>), 2000. ISBN: 0201704811
- FreeBSD: An Open-Source Operating System for Your Personal Computer (<http://andrsn.stanford.edu/FreeBSD/introbook/>), wydany przez The Bit Tree Press, 2001. ISBN: 0971204500
- Teach Yourself FreeBSD in 24 Hours, wydany przez Sams (<http://www.sampublishing.com/>), 2002. ISBN: 0672324245
- FreeBSD unleashed, wydany przez Sams (<http://www.sampublishing.com/>), 2002. ISBN: 0672324563
- FreeBSD: The Complete Reference, wydany przez McGrawHill (<http://books.mcgraw-hill.com>), 2003. ISBN: 0072224096

B.2. Podręczniki użytkownika

- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-075-9
- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-076-7
- *UNIX in a Nutshell*. O'Reilly & Associates, Inc., 1990. ISBN 093717520X
- Mui, Linda. *What You Need To Know When You Can't Find Your UNIX System Administrator*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-104-6
- Ohio State University (<http://www-wks.acs.ohio-state.edu/>) napisał kurs UNIX Introductory Course (http://www-wks.acs.ohio-state.edu/unix_course/unix.html), dostępny w sieci w formacie HTML i PostScript. Włoski przekład (http://www.FreeBSD.org/doc/it_IT.ISO8859-15/books/unix-introduction/index.html) tego dokumentu dostępny jest jako część projektu FreeBSD Italian Documentation Project.
- Jpman Project, Japan FreeBSD Users Group (<http://www.jp.FreeBSD.org/>). FreeBSD User's Reference Manual (<http://www.pc.mycom.co.jp/FreeBSD/urm.html>) (przekład japoński). Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>), 1998. ISBN4-8399-0088-4 P3800E.
- Edinburgh University (<http://www.ed.ac.uk/>) napisał przewodnik Online Guide (<http://unixhelp.ed.ac.uk/>) dla osób zaczynających pracę ze środowiskiem UNIX.

B.3. Podręczniki administracji

- Albitz, Paul and Liu, Cricket. *DNS and BIND*, 4th Ed. O'Reilly & Associates, Inc., 2001. ISBN 1-59600-158-4
- Computer Systems Research Group, UC Berkeley. *4.4BSD System Manager's Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-080-5
- Costales, Brian, et al. *Sendmail*, 2nd Ed. O'Reilly & Associates, Inc., 1997. ISBN 1-56592-222-0
- Frisch, Aileen. *Essential System Administration*, 2nd Ed. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-127-5
- Hunt, Craig. *TCP/IP Network Administration*, 2nd Ed. O'Reilly & Associates, Inc., 1997. ISBN 1-56592-322-7
- Nemeth, Evi. *UNIX System Administration Handbook*. 3rd Ed. Prentice Hall, 2000. ISBN 0-13-020601-6
- Stern, Hal *Managing NFS and NIS* O'Reilly & Associates, Inc., 1991. ISBN 0-937175-75-7
- Jpman Project, Japan FreeBSD Users Group (<http://www.jp.FreeBSD.org/>). FreeBSD System Administrator's Manual (<http://www.pc.mycom.co.jp/FreeBSD/sam.html>) (przekład japoński). Mainichi Communications Inc. (<http://www.pc.mycom.co.jp/>), 1998. ISBN4-8399-0109-0 P3300E.
- Dreyfus, Emmanuel. Cahiers de l'Admin: BSD (<http://www.eyrolles.com/Informatique/Livre/9782212114638/>) 2nd Ed. (po francusku), Eyrolles, 2004. ISBN 2-212-11463-X

B.4. Podręczniki programowania

- Asente, Paul, Converse, Diana, and Swick, Ralph. *X Window System Toolkit*. Digital Press, 1998. ISBN 1-55558-178-1
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-078-3
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-079-1
- Harbison, Samuel P. and Steele, Guy L. Jr. *C: A Reference Manual*. 4th ed. Prentice Hall, 1995. ISBN 0-13-326224-3
- Kernighan, Brian and Dennis M. Ritchie. *The C Programming Language*. 2nd Ed. PTR Prentice Hall, 1988. ISBN 0-13-110362-8
- Lehey, Greg. *Porting UNIX Software*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-126-7
- Plauger, P. J. *The Standard C Library*. Prentice Hall, 1992. ISBN 0-13-131509-9
- Spinellis, Diomidis. *Code Reading: The Open Source Perspective* (<http://www.spinellis.gr/codereading/>). Addison-Wesley, 2003. ISBN 0-201-79940-5
- Spinellis, Diomidis. *Code Quality: The Open Source Perspective* (<http://www.spinellis.gr/codequality/>). Addison-Wesley, 2006. ISBN 0-321-16607-8
- Stevens, W. Richard and Stephen A. Rago. *Advanced Programming in the UNIX Environment*. 2nd Ed. Reading, Mass. : Addison-Wesley, 2005. ISBN 0-201-43307-9
- Stevens, W. Richard. *UNIX Network Programming*. 2nd Ed, PTR Prentice Hall, 1998. ISBN 0-13-490012-X
- Wells, Bill. „Writing Serial Drivers for UNIX”. *Dr. Dobbs's Journal*. 19(15), December 1994. pp68-71, 97-99.

B.5. Komponenty systemu operacyjnego

- Andleigh, Prabhat K. *UNIX System Architecture*. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
 - Jolitz, William. „Porting UNIX to the 386”. *Dr. Dobbs's Journal*. January 1991-July 1992.
 - Leffler, Samuel J., Marshall Kirk McKusick, Michael J Karels i John Quarterman *The Design and Implementation of the 4.3BSD UNIX Operating System*. Reading, Mass. : Addison-Wesley, 1989. ISBN 0-201-06196-1
 - Leffler, Samuel J., Marshall Kirk McKusick, *The Design and Implementation of the 4.3BSD UNIX Operating System: Answer Book*. Reading, Mass. : Addison-Wesley, 1991. ISBN 0-201-54629-9
 - McKusick, Marshall Kirk, Keith Bostic, Michael J Karels i John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-54979-4
- (Rozdział 2 tej książki dostępny jest online (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/books/design-44bsd/book.html) jako część Projektu Dokumentacji FreeBSD, a rozdział 9 tutaj (http://www.netapp.com/tech_library/nfsbook.html).)
- Marshall Kirk McKusick, George V. Neville-Neil *The Design and Implementation of the FreeBSD Operating System*. Boston, Mass. : Addison-Wesley, 2004. ISBN 0-201-70245-2
 - Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63346-9
 - Schimmel, Curt. *Unix Systems for Modern Architectures*. Reading, Mass. : Addison-Wesley, 1994. ISBN 0-201-63338-8
 - Stevens, W. Richard. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63495-3
 - Vahalia, Uresh. *UNIX Internals -- The New Frontiers*. Prentice Hall, 1996. ISBN 0-13-101908-2
 - Wright, Gary R. i W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63354-X

B.6. Bezpieczeństwo

- Cheswick, William R. i Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63357-4
- Garfinkel, Simson i Gene Spafford. *Practical UNIX & Internet Security*. 2nd Ed. O'Reilly & Associates, Inc., 1996. ISBN 1-56592-148-8
- Garfinkel, Simson. *PGP Pretty Good Privacy* O'Reilly & Associates, Inc., 1995. ISBN 1-56592-098-8

B.7. Sprzęt

- Anderson, Don i Tom Shanley. *Pentium Processor System Architecture*. 2nd Ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40992-5

- Ferraro, Richard F. *Programmer's Guide to the EGA, VGA, and Super VGA Cards*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-62490-7
- Firma Intel Corporation publikuje dokumentacje ich procesorów, chipsetów i standardów na stronie projektantów (<http://developer.intel.com/>); z reguły jako pliki PDF.
- Shanley, Tom. *80486 System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40994-1
- Shanley, Tom. *ISA System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40996-8
- Shanley, Tom. *PCI System Architecture*. 4th ed. Reading, Mass. : Addison-Wesley, 1999. ISBN 0-201-30974-2
- Van GILLUWE, Frank. *The Undocumented PC*, 2nd Ed. Reading, Mass: Addison-Wesley Pub. Co., 1996. ISBN 0-201-47950-8
- Messmer, Hans-Peter. *The Indispensable PC Hardware Book*, 4th Ed. Reading, Mass: Addison-Wesley Pub. Co., 2002. ISBN 0-201-59616-4

B.8. Historia systemów UNIX

- Lion, John *Lion's Commentary on UNIX, 6th Ed. With Source Code*. ITP Media Group, 1996. ISBN 1573980137
- Raymond, Eric S. *The New Hacker's Dictionary, 3rd edition*. MIT Press, 1996. ISBN 0-262-68092-0. Znany również jako Jargon File (<http://www.catb.org/~esr/jargon/html/index.html>)
- Salus, Peter H. *A quarter century of UNIX*. Addison-Wesley Publishing Company, Inc., 1994. ISBN 0-201-54777-5
- Simon Garfinkel, Daniel Weise, Steven Strassmann. *The UNIX-HATERS Handbook*. IDG Books Worldwide, Inc., 1994. ISBN 1-56884-203-1. Nakład wyczerpany, lecz książka dostępna jest online (<http://research.microsoft.com/~daniel/unix-haters.html>).
- Don Libes, Sandy Ressler *Life with UNIX — special edition*. Prentice-Hall, Inc., 1989. ISBN 0-13-536657-7
- *The BSD family tree*. <http://www.FreeBSD.org/cgi/cvsweb.cgi/src/share/misc/bsd-family-tree> lub `/usr/share/misc/bsd-family-tree` w systemie FreeBSD.
- *The BSD Release Announcements collection*. 1997. <http://www.de.FreeBSD.org/de/ftp/releases/>
- *Networked Computer Science Technical Reports Library*. <http://www.ncstrl.org/>
- *Stare wydania BSD od Computer Systems Research group (CSRG)*. <http://www.mckusick.com/csrg/>: Zestaw 4 płyt CD zawiera wszystkie wersje BSD od 1BSD do 4.4BSD oraz 4.4BSD-Lite2 (lecz nie 2.11BSD, niestety). Ostatnia płyta zawiera również źródła wraz z plikami SCCS.

B.9. Czasopisma

- *The C/C++ Users Journal*. R&D Publications Inc. ISSN 1075-2838
- *Sys Admin — The Journal for UNIX System Administrators* Miller Freeman, Inc., ISSN 1061-2688
- *freeX — Das Magazin für Linux - BSD - UNIX* (in German) Computer- und Literaturverlag GmbH, ISSN 1436-7033

Dodatek C. Resources on the Internet

The rapid pace of FreeBSD progress makes print media impractical as a means of following the latest developments. Electronic resources are the best, if not often the only, way stay informed of the latest advances. Since FreeBSD is a volunteer effort, the user community itself also generally serves as a „technical support department” of sorts, with electronic mail and USENET news being the most effective way of reaching that community.

The most important points of contact with the FreeBSD user community are outlined below. If you are aware of other resources not mentioned here, please send them to the lista dyskusyjna projektu dokumentacji FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc>) so that they may also be included.

C.1. Mailing Lists

Though many of the FreeBSD development members read USENET, we cannot always guarantee that we will get to your questions in a timely fashion (or at all) if you post them only to one of the `comp.unix.bsd.freebsd.*` groups. By addressing your questions to the appropriate mailing list you will reach both us and a concentrated FreeBSD audience, invariably assuring a better (or at least faster) response.

The charters for the various lists are given at the bottom of this document. *Please read the charter before joining or sending mail to any list.* Most of our list subscribers now receive many hundreds of FreeBSD related messages every day, and by setting down charters and rules for proper use we are striving to keep the signal-to-noise ratio of the lists high. To do less would see the mailing lists ultimately fail as an effective communications medium for the project.

When in doubt about what list to post a question to, see How to get best results from the FreeBSD-questions mailing list (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/freebsd-questions).

Before posting to any list, please learn about how to best use the mailing lists, such as how to help avoid frequently-repeated discussions, by reading the Mailing List Frequently Asked Questions (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/mailling-list-faq) (FAQ) document.

Archives are kept for all of the mailing lists and can be searched using the FreeBSD World Wide Web server (<http://www.FreeBSD.org/search/index.html>). The keyword searchable archive offers an excellent way of finding answers to frequently asked questions and should be consulted before posting a question.

C.1.1. List Summary

General lists: The following are general lists which anyone is free (and encouraged) to join:

List	Purpose
cvs-all (http://lists.FreeBSD.org/mailman/listinfo/cvs-all)	Changes made to the FreeBSD source tree
freebsd-advocacy (http://lists.FreeBSD.org/mailman/listinfo/freebsd-advocacy)	FreeBSD Evangelism
freebsd-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce)	Important events and project milestones

List

freebsd-arch
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-arch>)

freebsd-bugbusters
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugbusters>)

freebsd-bugs
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugs>)

freebsd-chat
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat>)

freebsd-current
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>)

freebsd-isp
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp>)

freebsd-jobs
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-jobs>)

freebsd-policy
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-policy>)

freebsd-questions
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>)

freebsd-security-notifications
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications>)

freebsd-stable
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>)

freebsd-test
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-test>)

Purpose

Architecture and design discussions

Discussions pertaining to the maintenance of the FreeBSD problem report database and related tools

Bug reports

Non-technical items related to the FreeBSD community

Discussion concerning the use of FreeBSD-CURRENT

Issues for Internet Service Providers using FreeBSD

FreeBSD employment and consulting opportunities

FreeBSD Core team policy decisions. Low volume, and read-only

User questions and technical support

Security notifications

Discussion concerning the use of FreeBSD-STABLE

Where to send your test messages instead of one of the actual lists

Technical lists: The following lists are for technical discussion. You should read the charter for each list carefully before joining or sending mail to one as there are firm guidelines for their use and content.

List

freebsd-acpi
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>)

freebsd-afs
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-afs>)

freebsd-aic7xxx
(<http://lists.FreeBSD.org/mailman/listinfo/aic7xxx>)

Purpose

ACPI and power management development

Porting AFS to FreeBSD

Developing drivers for the Adaptec AIC 7xxx

List	Purpose
freebsd-alpha (http://lists.FreeBSD.org/mailman/listinfo/freebsd-alpha)	Porting FreeBSD to the Alpha
freebsd-amd64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-amd64)	Porting FreeBSD to AMD64 systems
freebsd-apache (http://lists.FreeBSD.org/mailman/listinfo/freebsd-apache)	Discussion about Apache related ports
freebsd-arm (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arm)	Porting FreeBSD to ARM® processors
freebsd-atm (http://lists.FreeBSD.org/mailman/listinfo/freebsd-atm)	Using ATM networking with FreeBSD
freebsd-audit (http://lists.FreeBSD.org/mailman/listinfo/freebsd-audit)	Source code audit project
freebsd-binup (http://lists.FreeBSD.org/mailman/listinfo/freebsd-binup)	Design and development of the binary update system
freebsd-bluetooth (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bluetooth)	Using Bluetooth technology in FreeBSD
freebsd-cluster (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cluster)	Using FreeBSD in a clustered environment
freebsd-cvsweb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cvsweb)	CVSweb maintenance
freebsd-database (http://lists.FreeBSD.org/mailman/listinfo/freebsd-database)	Discussing database use and development under FreeBSD
freebsd-doc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc)	Creating FreeBSD related documents
freebsd-drivers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-drivers)	Writing device drivers for FreeBSD
freebsd-eclipse (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eclipse)	FreeBSD users of Eclipse IDE, tools, rich client applications and ports.

List

freebsd-embedded
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded>)

freebsd-emulation
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-emulation>)

freebsd-firewire
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-firewire>)

freebsd-fs
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-fs>)

freebsd-geom
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-geom>)

freebsd-gnome
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-gnome>)

freebsd-hackers
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>)

freebsd-hardware
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hardware>)

freebsd-i18n
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-i18n>)

freebsd-ia32
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia32>)

freebsd-ia64
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia64>)

freebsd-ipfw
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ipfw>)

freebsd-isdn
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn>)

freebsd-java
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-java>)

freebsd-kde
(<http://freebsd.kde.org/mailman/listinfo/kde-freebsd>)

freebsd-lfs
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-lfs>)

Purpose

Using FreeBSD in embedded applications

Emulation of other systems such as
Linux/MS-DOS/Windows

FreeBSD FireWire® (iLink, IEEE 1394) technical
discussion

File systems

GEOM-specific discussions and implementations

Porting **GNOME** and **GNOME** applications

General technical discussion

General discussion of hardware for running FreeBSD

FreeBSD Internationalization

FreeBSD on the IA-32 (Intel x86) platform

Porting FreeBSD to Intel's upcoming IA64 systems

Technical discussion concerning the redesign of the IP
firewall code

ISDN developers

Java developers and people porting JDKs to FreeBSD

Porting **KDE** and **KDE** applications

Porting LFS to FreeBSD

List

freebsd-libh
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-libh>)

freebsd-mips
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-mips>)

freebsd-mobile
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-mobile>)

freebsd-mozilla
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-mozilla>)

freebsd-multimedia
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-multimedia>)

freebsd-new-bus
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-new-bus>)

freebsd-net
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-net>)

freebsd-openoffice
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-openoffice>)

freebsd-performance
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-performance>)

freebsd-perl
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-perl>)

freebsd-pf
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf>)

freebsd-platforms
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-platforms>)

freebsd-ports
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports>)

freebsd-ports-bugs
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs>)

Purpose

The second generation installation and package system

Porting FreeBSD to MIPS®

Discussions about mobile computing

Porting **Mozilla** to FreeBSD

Multimedia applications

Technical discussions about bus architecture

Networking discussion and TCP/IP source code

Porting **OpenOffice.org** and **StarOffice** to FreeBSD

Performance tuning questions for high performance/load installations

Maintenance of a number of Perl-related ports

Discussion and questions about the packet filter firewall system

Concerning ports to non Intel architecture platforms

Discussion of the Ports Collection

Discussion of the ports bugs/PRs

List

freebsd-ppc
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ppc>)

freebsd-proliant
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-proliant>)

freebsd-python
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-python>)

freebsd-qa
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-qa>)

freebsd-rc
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-rc>)

freebsd-realtime
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-realtime>)

freebsd-scsi
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-scsi>)

freebsd-security
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security>)

freebsd-small
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-small>)

freebsd-smp
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-smp>)

freebsd-sparc64
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-sparc64>)

freebsd-standards
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-standards>)

freebsd-threads
(<http://lists.FreeBSD.org/mailman/listinfo/freebsd-threads>)

Purpose

Porting FreeBSD to the PowerPC®

Technical discussion of FreeBSD on HP ProLiant server platforms

FreeBSD-specific Python issues

Discussion of Quality Assurance, usually pending a release

Discussion related to the `rc.d` system and its development

Development of realtime extensions to FreeBSD

The SCSI subsystem

Security issues affecting FreeBSD

Using FreeBSD in embedded applications (obsolete; use `freebsd-embedded` (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded>) instead)

Design discussions for [A]Symmetric MultiProcessing

Porting FreeBSD to SPARC® based systems

FreeBSD's conformance to the C99 and the POSIX standards

Threading in FreeBSD

List	Purpose
freebsd-testing (http://lists.FreeBSD.org/mailman/listinfo/freebsd-testing)	FreeBSD Performance and Stability Tests
freebsd-tokenring (http://lists.FreeBSD.org/mailman/listinfo/freebsd-tokenring)	Support Token Ring in FreeBSD
freebsd-usb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-usb)	Discussing FreeBSD support for USB
freebsd-vuxml (http://lists.FreeBSD.org/mailman/listinfo/freebsd-vuxml)	Discussion on VuXML infrastructure
freebsd-x11 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-x11)	Maintenance and support of X11 on FreeBSD

Limited lists: The following lists are for more specialized (and demanding) audiences and are probably not of interest to the general public. It is also a good idea to establish a presence in the technical lists before joining one of these limited lists so that you will understand the communications etiquette involved.

List	Purpose
freebsd-hubs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hubs)	People running mirror sites (infrastructural support)
freebsd-user-groups (http://lists.FreeBSD.org/mailman/listinfo/freebsd-user-groups)	User group coordination
freebsd-vendors (http://lists.FreeBSD.org/mailman/listinfo/freebsd-vendors)	Vendors pre-release coordination
freebsd-www (http://lists.FreeBSD.org/mailman/listinfo/freebsd-www)	Maintainers of www.FreeBSD.org (http://www.FreeBSD.org/index.html)

Digest lists: All of the above lists are available in a digest format. Once subscribed to a list, you can change your digest options in your account options section.

CVS lists: The following lists are for people interested in seeing the log messages for changes to various areas of the source tree. They are *Read-Only* lists and should not have mail sent to them.

List	Source area	Area Description (source for)
cvs-all (http://lists.FreeBSD.org/mailman/listinfo/cvs-all)	/usr / (CVSROOT doc ports project release src)	All changes to any place in the tree (superset of other CVS commit lists)

List	Source area	Area Description (source for)
cvs-doc (http://lists.FreeBSD.org/mailman/listinfo/cvs-doc)	<code>/usr/(doc www)</code>	All changes to the doc and www trees
cvs-ports (http://lists.FreeBSD.org/mailman/listinfo/cvs-ports)	<code>/usr/ports</code>	All changes to the ports tree
cvs-projects (http://lists.FreeBSD.org/mailman/listinfo/cvs-projects)	<code>/usr/projects</code>	All changes to the projects tree
cvs-src (http://lists.FreeBSD.org/mailman/listinfo/cvs-src)	<code>/usr/src</code>	All changes to the src tree

C.1.2. How to Subscribe

To subscribe to a list, click on the list name above or go to <http://lists.FreeBSD.org/mailman/listinfo> and click on the list that you are interested in. The list page should contain all of the necessary subscription instructions.

To actually post to a given list you simply send mail to `<listname@FreeBSD.org>`. It will then be redistributed to mailing list members world-wide.

To unsubscribe yourself from a list, click on the URL found at the bottom of every email received from the list. It is also possible to send an email to `<listname-unsubscribe@FreeBSD.org>` to unsubscribe yourself.

Again, we would like to request that you keep discussion in the technical mailing lists on a technical track. If you are only interested in important announcements then it is suggested that you join the Informacyjna lista dyskusyjna FreeBSD (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce>), which is intended only for infrequent traffic.

C.1.3. List Charters

All FreeBSD mailing lists have certain basic rules which must be adhered to by anyone using them. Failure to comply with these guidelines will result in two (2) written warnings from the FreeBSD Postmaster `<postmaster@FreeBSD.org>`, after which, on a third offense, the poster will be removed from all FreeBSD mailing lists and filtered from further posting to them. We regret that such rules and measures are necessary at all, but today's Internet is a pretty harsh environment, it would seem, and many fail to appreciate just how fragile some of its mechanisms are.

Rules of the road:

- The topic of any posting should adhere to the basic charter of the list it is posted to, e.g. if the list is about technical issues then your posting should contain technical discussion. Ongoing irrelevant chatter or flaming only detracts from the value of the mailing list for everyone on it and will not be tolerated. For free-form discussion on

no particular topic, the FreeBSD chat mailing list (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat>) is freely available and should be used instead.

- No posting should be made to more than 2 mailing lists, and only to 2 when a clear and obvious need to post to both lists exists. For most lists, there is already a great deal of subscriber overlap and except for the most esoteric mixes (say „-stable & -scsi”), there really is no reason to post to more than one list at a time. If a message is sent to you in such a way that multiple mailing lists appear on the Cc line then the Cc line should also be trimmed before sending it out again. *You are still responsible for your own cross-postings, no matter who the originator might have been.*
- Personal attacks and profanity (in the context of an argument) are not allowed, and that includes users and developers alike. Gross breaches of netiquette, like excerpting or reposting private mail when permission to do so was not and would not be forthcoming, are frowned upon but not specifically enforced. *However*, there are also very few cases where such content would fit within the charter of a list and it would therefore probably rate a warning (or ban) on that basis alone.
- Advertising of non-FreeBSD related products or services is strictly prohibited and will result in an immediate ban if it is clear that the offender is advertising by spam.

Individual list charters:

freebsd-acpi (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi>)

ACPI and power management development

freebsd-afs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-afs>)

Andrew File System

This list is for discussion on porting and using AFS from CMU/Transarc

freebsd-announce (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce>)

Important events / milestones

This is the mailing list for people interested only in occasional announcements of significant FreeBSD events. This includes announcements about snapshots and other releases. It contains announcements of new FreeBSD capabilities. It may contain calls for volunteers etc. This is a low volume, strictly moderated mailing list.

freebsd-arch (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-arch>)

Architecture and design discussions

This list is for discussion of the FreeBSD architecture. Messages will mostly be kept strictly technical in nature. Examples of suitable topics are:

- How to re-vamp the build system to have several customized builds running at the same time.
- What needs to be fixed with VFS to make Heidemann layers work.
- How do we change the device driver interface to be able to use the same drivers cleanly on many buses and architectures.
- How to write a network driver.

freebsd-audit (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-audit>)

Source code audit project

This is the mailing list for the FreeBSD source code audit project. Although this was originally intended for security-related changes, its charter has been expanded to review any code changes.

This list is very heavy on patches, and is probably of no interest to the average FreeBSD user. Security discussions not related to a particular code change are held on freebsd-security. Conversely, all developers are encouraged to send their patches here for review, especially if they touch a part of the system where a bug may adversely affect the integrity of the system.

freebsd-binup (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-binup>)

FreeBSD Binary Update Project

This list exists to provide discussion for the binary update system, or **binup**. Design issues, implementation details, patches, bug reports, status reports, feature requests, commit logs, and all other things related to **binup** are fair game.

freebsd-bluetooth (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-bluetooth>)

Bluetooth in FreeBSD

This is the forum where FreeBSD's Bluetooth users congregate. Design issues, implementation details, patches, bug reports, status reports, feature requests, and all matters related to Bluetooth are fair game.

freebsd-bugbusters (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugbusters>)

Coordination of the Problem Report handling effort

The purpose of this list is to serve as a coordination and discussion forum for the Bugmeister, his Bugbusters, and any other parties who have a genuine interest in the PR database. This list is not for discussions about specific bugs, patches or PRs.

freebsd-bugs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugs>)

Bug reports

This is the mailing list for reporting bugs in FreeBSD. Whenever possible, bugs should be submitted using the send-pr(1) command or the WEB interface (<http://www.FreeBSD.org/send-pr.html>) to it.

freebsd-chat (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat>)

Non technical items related to the FreeBSD community

This list contains the overflow from the other lists about non-technical, social information. It includes discussion about whether Jordan looks like a toon ferret or not, whether or not to type in capitals, who is drinking too much coffee, where the best beer is brewed, who is brewing beer in their basement, and so on. Occasional announcements of important events (such as upcoming parties, weddings, births, new jobs, etc) can be made to the technical lists, but the follow ups should be directed to this -chat list.

freebsd-core

FreeBSD core team

This is an internal mailing list for use by the core members. Messages can be sent to it when a serious FreeBSD-related matter requires arbitration or high-level scrutiny.

freebsd-current (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-current>)

Discussions about the use of FreeBSD-CURRENT

This is the mailing list for users of FreeBSD-CURRENT. It includes warnings about new features coming out in -CURRENT that will affect the users, and instructions on steps that must be taken to remain -CURRENT. Anyone running „CURRENT” must subscribe to this list. This is a technical mailing list for which strictly technical content is expected.

freebsd-cvsweb (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-cvsweb>)

FreeBSD CVSweb Project

Technical discussions about use, development and maintenance of FreeBSD-CVSweb.

freebsd-doc (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc>)

Documentation project

This mailing list is for the discussion of issues and projects related to the creation of documentation for FreeBSD. The members of this mailing list are collectively referred to as „The FreeBSD Documentation Project”. It is an open list; feel free to join and contribute!

freebsd-drivers (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-drivers>)

Writing device drivers for FreeBSD

This is a forum for technical discussions related to device drivers on FreeBSD. It is primarily a place for device driver writers to ask questions about how to write device drivers using the APIs in the FreeBSD kernel.

freebsd-eclipse (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-eclipse>)

FreeBSD users of Eclipse IDE, tools, rich client applications and ports.

The intention of this list is to provide mutual support for everything to do with choosing, installing, using, developing and maintaining the Eclipse IDE, tools, rich client applications on the FreeBSD platform and assisting with the porting of Eclipse IDE and plugins to the FreeBSD environment.

The intention is also to facilitate exchange of information between the Eclipse community and the FreeBSD community to the mutual benefit of both.

Although this list is focused primarily on the needs of Eclipse users it will also provide a forum for those who would like to develop FreeBSD specific applications using the Eclipse framework.

freebsd-embedded (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded>)

Using FreeBSD in embedded applications

This list discusses topics related to using FreeBSD in embedded systems. This is a technical mailing list for which strictly technical content is expected. For the purpose of this list we define embedded systems as those computing devices which are not desktops and which usually serve a single purpose as opposed to being general computing environments. Examples include, but are not limited to, all kinds of phone handsets, network equipment such as routers, switches and PBXs, remote measuring equipment, PDAs, Point Of Sale systems, and so on.

freebsd-emulation (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-emulation>)

Emulation of other systems such as Linux/MS-DOS/Windows

This is a forum for technical discussions related to running programs written for other operating systems on FreeBSD.

freebsd-firewire (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-firewire>)

FireWire (iLink, IEEE 1394)

This is a mailing list for discussion of the design and implementation of a FireWire (aka IEEE 1394 aka iLink) subsystem for FreeBSD. Relevant topics specifically include the standards, bus devices and their protocols, adapter boards/cards/chips sets, and the architecture and implementation of code for their proper support.

freebsd-fs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-fs>)

File systems

Discussions concerning FreeBSD file systems. This is a technical mailing list for which strictly technical content is expected.

freebsd-geom (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-geom>)

GEOM

Discussions specific to GEOM and related implementations. This is a technical mailing list for which strictly technical content is expected.

freebsd-gnome (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-gnome>)

GNOME

Discussions concerning The **GNOME** Desktop Environment for FreeBSD systems. This is a technical mailing list for which strictly technical content is expected.

freebsd-ipfw (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ipfw>)

IP Firewall

This is the forum for technical discussions concerning the redesign of the IP firewall code in FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-ia64 (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia64>)

Porting FreeBSD to IA64

This is a technical mailing list for individuals actively working on porting FreeBSD to the IA-64 platform from Intel, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome.

freebsd-isdn (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn>)

ISDN Communications

This is the mailing list for people discussing the development of ISDN support for FreeBSD.

freebsd-java (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-java>)

Java Development

This is the mailing list for people discussing the development of significant Java applications for FreeBSD and the porting and maintenance of JDKs.

freebsd-jobs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-jobs>)

Jobs offered and sought

This is a forum for posting employment notices and resumes specifically related to FreeBSD, e.g. if you are seeking FreeBSD-related employment or have a job involving FreeBSD to advertise then this is the right place. This is *not* a mailing list for general employment issues since adequate forums for that already exist elsewhere.

Note that this list, like other `FreeBSD.org` mailing lists, is distributed worldwide. Thus, you need to be clear about location and the extent to which telecommuting or assistance with relocation is available.

Email should use open formats only — preferably plain text, but basic Portable Document Format (PDF), HTML, and a few others are acceptable to many readers. Closed formats such as Microsoft Word (`.doc`) will be rejected by the mailing list server.

freebsd-kde (<http://freebsd.kde.org/mailman/listinfo/kde-freebsd>)

KDE

Discussions concerning **KDE** on FreeBSD systems. This is a technical mailing list for which strictly technical content is expected.

freebsd-hackers (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers>)

Technical discussions

This is a forum for technical discussions related to FreeBSD. This is the primary technical mailing list. It is for individuals actively working on FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome. This is a technical mailing list for which strictly technical content is expected.

freebsd-hardware (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hardware>)

General discussion of FreeBSD hardware

General discussion about the types of hardware that FreeBSD runs on, various problems and suggestions concerning what to buy or avoid.

freebsd-hubs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-hubs>)

Mirror sites

Announcements and discussion for people who run FreeBSD mirror sites.

freebsd-isp (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp>)

Issues for Internet Service Providers

This mailing list is for discussing topics relevant to Internet Service Providers (ISPs) using FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-openoffice (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-openoffice>)

OpenOffice.org

Discussions concerning the porting and maintenance of **OpenOffice.org** and **StarOffice**.

freebsd-performance (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-performance>)

Discussions about tuning or speeding up FreeBSD

This mailing list exists to provide a place for hackers, administrators, and/or concerned parties to discuss performance related topics pertaining to FreeBSD. Acceptable topics includes talking about FreeBSD installations that are either under high load, are experiencing performance problems, or are pushing the limits of FreeBSD. Concerned parties that are willing to work toward improving the performance of FreeBSD are highly encouraged to subscribe to this list. This is a highly technical list ideally suited for experienced FreeBSD users, hackers, or administrators interested in keeping FreeBSD fast, robust, and scalable. This list is not a question-and-answer list that replaces reading through documentation, but it is a place to make contributions or inquire about unanswered performance related topics.

freebsd-pf (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf>)

Discussion and questions about the packet filter firewall system

Discussion concerning the packet filter (pf) firewall system in terms of FreeBSD. Technical discussion and user questions are both welcome. This list is also a place to discuss the ALTQ QoS framework.

freebsd-platforms (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-platforms>)

Porting to Non Intel platforms

Cross-platform FreeBSD issues, general discussion and proposals for non Intel FreeBSD ports. This is a technical mailing list for which strictly technical content is expected.

freebsd-policy (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-policy>)

Core team policy decisions

This is a low volume, read-only mailing list for FreeBSD Core Team Policy decisions.

freebsd-ports (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports>)

Discussion of „ports”

Discussions concerning FreeBSD's „ports collection” (`/usr/ports`), ports infrastructure, and general ports coordination efforts. This is a technical mailing list for which strictly technical content is expected.

freebsd-ports-bugs (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs>)

Discussion of „ports” bugs

Discussions concerning problem reports for FreeBSD's „ports collection” (`/usr/ports`), proposed ports, or modifications to ports. This is a technical mailing list for which strictly technical content is expected.

freebsd-proliant (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-proliant>)

Technical discussion of FreeBSD on HP ProLiant server platforms

This mailing list is to be used for the technical discussion of the usage of FreeBSD on HP ProLiant servers, including the discussion of ProLiant-specific drivers, management software, configuration tools, and BIOS updates. As such, this is the primary place to discuss the `hpsmd`, `hpasmcli`, and `hpacucli` modules.

freebsd-python (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-python>)

Python on FreeBSD

This is a list for discussions related to improving Python-support on FreeBSD. This is a technical mailing list. It is for individuals working on porting Python, its 3rd party modules and **Zope** stuff to FreeBSD. Individuals interested in following the technical discussion are also welcome.

freebsd-questions (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions>)

User questions

This is the mailing list for questions about FreeBSD. You should not send „how to” questions to the technical lists unless you consider the question to be pretty technical.

freebsd-scsi (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-scsi>)

SCSI subsystem

This is the mailing list for people working on the SCSI subsystem for FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-security (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security>)

Security issues

FreeBSD computer security issues (DES, Kerberos, known security holes and fixes, etc). This is a technical mailing list for which strictly technical discussion is expected. Note that this is not a question-and-answer list, but that contributions (BOTH question AND answer) to the FAQ are welcome.

freebsd-security-notifications (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications>)

Security Notifications

Notifications of FreeBSD security problems and fixes. This is not a discussion list. The discussion list is FreeBSD-security.

freebsd-small (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-small>)

Using FreeBSD in embedded applications

This list discusses topics related to unusually small and embedded FreeBSD installations. This is a technical mailing list for which strictly technical content is expected.

Notatka: This list has been obsoleted by freebsd-embedded (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded>).

freebsd-stable (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable>)

Discussions about the use of FreeBSD-STABLE

This is the mailing list for users of FreeBSD-STABLE. It includes warnings about new features coming out in -STABLE that will affect the users, and instructions on steps that must be taken to remain -STABLE. Anyone running „STABLE” should subscribe to this list. This is a technical mailing list for which strictly technical content is expected.

freebsd-standards (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-standards>)

C99 & POSIX Conformance

This is a forum for technical discussions related to FreeBSD Conformance to the C99 and the POSIX standards.

freebsd-usb (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-usb>)

Discussing FreeBSD support for USB

This is a mailing list for technical discussions related to FreeBSD support for USB.

freebsd-user-groups (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-user-groups>)

User Group Coordination List

This is the mailing list for the coordinators from each of the local area Users Groups to discuss matters with each other and a designated individual from the Core Team. This mail list should be limited to meeting synopsis and coordination of projects that span User Groups.

freebsd-vendors (<http://lists.FreeBSD.org/mailman/listinfo/freebsd-vendors>)

Vendors

Coordination discussions between The FreeBSD Project and Vendors of software and hardware for FreeBSD.

C.1.4. Filtering on the Mailing Lists

The FreeBSD mailing lists are filtered in multiple ways to avoid the distribution of spam, viruses, and other unwanted emails. The filtering actions described in this section do not include all those used to protect the mailing lists.

Only certain types of attachments are allowed on the mailing lists. All attachments with a MIME content type not found in the list below will be stripped before an email is distributed on the mailing lists.

- application/octet-stream
- application/pdf
- application/pgp-signature
- application/x-pkcs7-signature
- message/rfc822
- multipart/alternative
- multipart/related
- multipart/signed
- text/html
- text/plain
- text/x-diff
- text/x-patch

Notatka: Some of the mailing lists might allow attachments of other MIME content types, but the above list should be applicable for most of the mailing lists.

If an email contains both an HTML and a plain text version, the HTML version will be removed. If an email contains only an HTML version, it will be converted to plain text.

C.2. Usenet Newsgroups

In addition to two FreeBSD specific newsgroups, there are many others in which FreeBSD is discussed or are otherwise relevant to FreeBSD users. Keyword searchable archives (http://minnie.tuhs.org/BSD-info/bsdnews_search.html) are available for some of these newsgroups from courtesy of Warren Toomey <wkt@cs.adfa.edu.au>.

C.2.1. BSD Specific Newsgroups

- comp.unix.bsd.freebsd.announce (news:comp.unix.bsd.freebsd.announce)
- comp.unix.bsd.freebsd.misc (news:comp.unix.bsd.freebsd.misc)
- de.comp.os.unix.bsd (news:de.comp.os.unix.bsd) (German)
- fr.comp.os.bsd (news:fr.comp.os.bsd) (French)
- it.comp.os.freebsd (news:it.comp.os.freebsd) (Italian)

C.2.2. Other UNIX Newsgroups of Interest

- comp.unix (news:comp.unix)
- comp.unix.questions (news:comp.unix.questions)
- comp.unix.admin (news:comp.unix.admin)
- comp.unix.programmer (news:comp.unix.programmer)
- comp.unix.shell (news:comp.unix.shell)
- comp.unix.user-friendly (news:comp.unix.user-friendly)
- comp.security.unix (news:comp.security.unix)
- comp.sources.unix (news:comp.sources.unix)
- comp.unix.advocacy (news:comp.unix.advocacy)
- comp.unix.misc (news:comp.unix.misc)
- comp.bugs.4bsd (news:comp.bugs.4bsd)
- comp.bugs.4bsd.ucb-fixes (news:comp.bugs.4bsd.ucb-fixes)
- comp.unix.bsd (news:comp.unix.bsd)

C.2.3. X Window System

- comp.windows.x.i386unix (news:comp.windows.x.i386unix)
- comp.windows.x (news:comp.windows.x)
- comp.windows.x.apps (news:comp.windows.x.apps)
- comp.windows.x.announce (news:comp.windows.x.announce)

- `comp.windows.x.intrinsics` (`news:comp.windows.x.intrinsics`)
- `comp.windows.x.motif` (`news:comp.windows.x.motif`)
- `comp.windows.x.pex` (`news:comp.windows.x.pex`)
- `comp.emulators.ms-windows.wine` (`news:comp.emulators.ms-windows.wine`)

C.3. World Wide Web Servers

Central Servers, Armenia, Australia, Austria, Belgium, Brazil, Canada, China, Costa Rica, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hong Kong, Hungary, Iceland, Italy, Japan, Korea, Kuwait, Kyrgyzstan, Latvia, Lithuania, Netherlands, Norway, Philippines, Portugal, Romania, Russia, San Marino, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Thailand, Turkey, Ukraine, United Kingdom, USA.

(as of UTC)

- Central Servers
 - <http://www.FreeBSD.org/>
- Armenia
 - <http://www1.am.FreeBSD.org/> (IPv6)
- Australia
 - <http://www.au.FreeBSD.org/>
 - <http://www2.au.FreeBSD.org/>
- Austria
 - <http://www.at.FreeBSD.org/> (IPv6)
- Belgium
 - <http://freebsd.unixtech.be/>
-

Brazil

- <http://www.br.FreeBSD.org/> (IPv6)
- <http://www2.br.FreeBSD.org/www.freebsd.org/>
- <http://www3.br.FreeBSD.org/>

•

Canada

- <http://www.ca.FreeBSD.org/>
- <http://www2.ca.FreeBSD.org/>

•

China

- <http://www.cn.FreeBSD.org/>

•

Costa Rica

- <http://www1.cr.FreeBSD.org/>

•

Czech Republic

- <http://www.cz.FreeBSD.org/> (IPv6)

•

Denmark

- <http://www.dk.FreeBSD.org/> (IPv6)

•

Estonia

- <http://www.ee.FreeBSD.org/>

•

Finland

- <http://www.fi.FreeBSD.org/>
- <http://www2.fi.FreeBSD.org/>

- France
 - <http://www.fr.FreeBSD.org/>
 - <http://www1.fr.FreeBSD.org/>
- Germany
 - <http://www.de.FreeBSD.org/>
- Greece
 - <http://www.gr.FreeBSD.org/>
- Hong Kong
 - <http://www.hk.FreeBSD.org/>
- Hungary
 - <http://www.hu.FreeBSD.org/>
 - <http://www2.hu.FreeBSD.org/>
- Iceland
 - <http://www.is.FreeBSD.org/>
- Italy
 - <http://www.it.FreeBSD.org/>
 - <http://www.gufi.org/mirrors/www.freebsd.org/data/>
- Japan
 - <http://www.jp.FreeBSD.org/www.FreeBSD.org/> (IPv6)

- Korea
 - <http://www.kr.FreeBSD.org/>
 - <http://www2.kr.FreeBSD.org/>
- Kuwait
 - <http://www.kw.FreeBSD.org/>
- Kyrgyzstan
 - <http://www.kg.FreeBSD.org/>
- Latvia
 - <http://www.lv.FreeBSD.org/>
 - <http://www2.lv.FreeBSD.org/>
- Lithuania
 - <http://www.lt.FreeBSD.org/>
- Netherlands
 - <http://www.nl.FreeBSD.org/>
 - <http://www2.nl.FreeBSD.org/>
- Norway
 - <http://www.no.FreeBSD.org/>
- Philippines
 - <http://www.FreeBSD.org.ph/>

•

Portugal

- <http://www.pt.FreeBSD.org/>
- <http://www1.pt.FreeBSD.org/>
- <http://www4.pt.FreeBSD.org/>
- <http://www5.pt.FreeBSD.org/>

•

Romania

- <http://www.ro.FreeBSD.org/>
- <http://www1.ro.FreeBSD.org/>
- <http://www2.ro.FreeBSD.org/>
- <http://www3.ro.FreeBSD.org/>

•

Russia

- <http://www.ru.FreeBSD.org/>
- <http://www2.ru.FreeBSD.org/>
- <http://www3.ru.FreeBSD.org/>
- <http://www4.ru.FreeBSD.org/>
- <http://www5.ru.FreeBSD.org/>

•

San Marino

- <http://www.sm.FreeBSD.org/>

•

Slovak Republic

- <http://www.sk.FreeBSD.org/>

•

Slovenia

- <http://www.si.FreeBSD.org/>
- <http://www2.si.FreeBSD.org/>

•

South Africa

- <http://www.za.FreeBSD.org/>
- <http://www2.za.FreeBSD.org/>

•

Spain

- <http://www.es.FreeBSD.org/>
- <http://www2.es.FreeBSD.org/>
- <http://www3.es.FreeBSD.org/>

•

Sweden

- <http://www.se.FreeBSD.org/>
- <http://www2.se.FreeBSD.org/>

•

Switzerland

- <http://www.ch.FreeBSD.org/> (IPv6)
- <http://www2.ch.FreeBSD.org/> (IPv6)

•

Taiwan

- <http://www.tw.FreeBSD.org/> (IPv6)
- <http://www2.tw.FreeBSD.org/>
- <http://www3.tw.FreeBSD.org/>
- <http://www4.tw.FreeBSD.org/>
- <http://www5.tw.FreeBSD.org/> (IPv6)
- <http://www6.tw.FreeBSD.org/>
- <http://www7.tw.FreeBSD.org/>

•

Thailand

- <http://www.th.FreeBSD.org/>

•

Turkey

- <http://www.tr.FreeBSD.org/>
- <http://www2.tr.FreeBSD.org/>
- <http://www3.tr.FreeBSD.org/> (IPv6)

•

Ukraine

- <http://www.ua.FreeBSD.org/>
- <http://www2.ua.FreeBSD.org/>
- <http://www5.ua.FreeBSD.org/>
- <http://www4.ua.FreeBSD.org/>

•

United Kingdom

- <http://www1.uk.FreeBSD.org/>
- <http://www3.uk.FreeBSD.org/>

•

USA

- <http://www2.us.FreeBSD.org/>
- <http://www5.us.FreeBSD.org/> (IPv6)

C.4. Email Addresses

The following user groups provide FreeBSD related email addresses for their members. The listed administrator reserves the right to revoke the address if it is abused in any way.

Domain	Facilities	User Group	Administrator
ukug.uk.FreeBSD.org	Forwarding only	<freebsd-users@uk.FreeBSD.org>	Lee Johnston <lee@uk.FreeBSD.org>

C.5. Shell Accounts

The following user groups provide shell accounts for people who are actively supporting the FreeBSD project. The

listed administrator reserves the right to cancel the account if it is abused in any way.

Host	Access	Facilities	Administrator
dogma.freebsd-uk.eu.org	Telnet/FTP/SSH	Email, Web space, Anonymous FTP	Lee Johnston <lee@uk.FreeBSD.org>

Dodatek D. Klucze PGP

W tym rozdziale zostało zebranych, dla naszej wygody, wiele kluczy oficerów czy twórców FreeBSD, gdybyśmy musieli zweryfikować podpis bądź wysłać do jednego z nich zaszyfrowaną wiadomość. Kompletna baza kluczy użytkowników FreeBSD.org dostępna jest pod adresem <http://www.FreeBSD.org/doc/pgpkeyring.txt>.

D.1. Oficerowie

D.1.1. Security Officer Team <security-officer@FreeBSD.org>

```
pub 1024D/CA6CDFB2 2002-08-27 FreeBSD Security Officer <security-officer@FreeBSD.org>
    Key fingerprint = C374 0FC5 69A6 FBB1 4AED B131 15D6 8804 CA6C DFB2
sub 2048g/A3071809 2002-08-27
```

D.1.2. Core Team Secretary <core-secretary@FreeBSD.org>

```
pub 2048R/2CA49776 2012-07-23
    Key fingerprint = 89F6 C031 B4E3 D472 E4CE 8372 4D58 FDCD 2CA4 9776
uid FreeBSD Core Team Secretary <core-secretary@freebsd.org>
sub 2048R/BBAD1C98 2012-07-23
```

D.1.3. Ports Management Team Secretary <portmgr-secretary@FreeBSD.org>

```
pub 1024D/7414629C 2005-11-30
    Key fingerprint = D50C BA61 8DC6 C42E 4C05 BF9A 79F6 E071 7414 629C
uid FreeBSD portmgr secretary <portmgr-secretary@FreeBSD.org>
sub 2048g/80B696E6 2005-11-30
```

D.2. Członkowie głównego zespołu projektantów

D.2.1. Thomas Abthorpe <tabthorpe@FreeBSD.org>

```
pub 2048R/A473C990 2010-05-28
    Key fingerprint = D883 2D7C EB78 944A 69FC 36A6 D937 1097 A473 C990
uid Thomas Abthorpe (FreeBSD Committer) <tabthorpe@FreeBSD.org>
uid Thomas Abthorpe <thomas@goodking.ca>
uid Thomas Abthorpe <tabthorpe@goodking.org>
sub 2048R/8CA60EE0 2010-05-28
```

D.2.2. Gavin Atkinson <gavin@FreeBSD.org>

```
pub 1024D/A093262B 2005-02-18
    Key fingerprint = 313A A79F 697D 3A5C 216A EDF5 935D EF44 A093 262B
uid          Gavin Atkinson <gavin@16squared.co.uk>
uid          Gavin Atkinson (FreeBSD key) <gavin@FreeBSD.org>
uid          Gavin Atkinson (Work e-mail) <ga9@york.ac.uk>
uid          Gavin Atkinson <gavin.atkinson@ury.york.ac.uk>
sub 2048g/58F40B3D 2005-02-18
```

D.2.3. John Baldwin <jhb@FreeBSD.org>

```
pub 1024R/C10A874D 1999-01-13 John Baldwin <jbaldwin@weather.com>
    Key fingerprint = 43 33 1D 37 72 B1 EF 5B 9B 5F 39 F8 BD C1 7C B5
uid          John Baldwin <john@baldwin.cx>
uid          John Baldwin <jhb@FreeBSD.org>
uid          John Baldwin <jobaldwi@vt.edu>
```

D.2.4. Konstantin Belousov <kib@FreeBSD.org>

```
pub 1024D/DD4C6F88 2004-07-29
    Key fingerprint = 39DA E615 A45C 111D 777B 3AD0 0B7F 8C04 DD4C 6F88
uid          Konstantin Belousov <kib@freebsd.org>
uid          Konstantin Belousov <konstantin.belousov@zoral.com.ua>
uid          Kostik Belousov <kostikbel@ukr.net>
uid          Kostik Belousov <kostikbel@gmail.com>
sub 2048g/18488597 2004-07-29
```

D.2.5. David Chisnall <theraven@FreeBSD.org>

```
pub 2048R/4D302EBF 2011-11-06
    Key fingerprint = 650B 1F1D 20F7 F139 77BD 25C4 E07B 6A41 4D30 2EBF
uid          David Chisnall <theraven@FreeBSD.org>
uid          David Chisnall <theraven@theravensnest.org>
sub 2048R/C6103EC2 2011-11-06
```

D.2.6. Hiroki Sato <hrs@FreeBSD.org>

```
pub 1024D/2793CF2D 2001-06-12
    Key fingerprint = BDB3 443F A5DD B3D0 A530 FFD7 4F2C D3D8 2793 CF2D
uid          Hiroki Sato <hrs@allbsd.org>
uid          Hiroki Sato <hrs@eos.ocn.ne.jp>
uid          Hiroki Sato <hrs@ring.gr.jp>
uid          Hiroki Sato <hrs@FreeBSD.org>
uid          Hiroki Sato <hrs@jp.FreeBSD.org>
uid          Hiroki Sato <hrs@vlsi.ee.noda.tus.ac.jp>
```

```
uid          Hiroki Sato <hrs@jp.NetBSD.org>
uid          Hiroki Sato <hrs@NetBSD.org>
uid          Hiroki Sato <hrs@ec.ss.titech.ac.jp>
uid          Hiroki Sato <hrs@ieee.org>
uid          Hiroki Sato <hrs@acm.org>
sub 1024g/8CD251FF 2001-06-12
```

D.2.7. Peter Wemm <peter@FreeBSD.org>

```
pub 1024D/7277717F 2003-12-14 Peter Wemm <peter@wemm.org>
   Key fingerprint = 622B 2282 E92B 3BAB 57D1 A417 1512 AE52 7277 717F
uid          Peter Wemm <peter@FreeBSD.ORG>
sub 1024g/8B40D9D1 2003-12-14
pub 1024R/D89CE319 1995-04-02 Peter Wemm <peter@netplex.com.au>
   Key fingerprint = 47 05 04 CA 4C EE F8 93 F6 DB 02 92 6D F5 58 8A
uid          Peter Wemm <peter@perth.dialix.oz.au>
uid          Peter Wemm <peter@haywire.dialix.com>
```

D.2.8. Martin Wilke <miwi@FreeBSD.org>

```
pub 1024D/B1E6FCE9 2009-01-31
   Key fingerprint = C022 7D60 F598 8188 2635 0F6E 74B2 4884 B1E6 FCE9
uid          Martin Wilke <miwi@FreeBSD.org>
sub 4096g/096DA69D 2009-01-31
```

D.3. Twórcy

D.3.1. Ariff Abdullah <ariff@FreeBSD.org>

```
pub 1024D/C5304CDA 2005-10-01
   Key fingerprint = 5C7C 6BF4 8293 DE76 27D9 FD57 96BF 9D78 C530 4CDA
uid          Ariff Abdullah <skywizard@MyBSD.org.my>
uid          Ariff Abdullah <ariff@MyBSD.org.my>
uid          Ariff Abdullah <ariff@FreeBSD.org>
sub 2048g/8958C1D3 2005-10-01
```

D.3.2. Thomas Abthorpe <tabthorpe@FreeBSD.org>

```
pub 2048R/A473C990 2010-05-28
   Key fingerprint = D883 2D7C EB78 944A 69FC 36A6 D937 1097 A473 C990
uid          Thomas Abthorpe (FreeBSD Committer) <tabthorpe@FreeBSD.org>
uid          Thomas Abthorpe <thomas@goodking.ca>
uid          Thomas Abthorpe <tabthorpe@goodking.org>
sub 2048R/8CA60EE0 2010-05-28
```

D.3.3. Eitan Adler <eadler@FreeBSD.org>

```
pub 4096R/8FC8196C 2011-02-11
    Key fingerprint = 49C7 29DF E09C 0FC7 A1C4 6ECB A338 A6FC 8FC8 196C
uid          Eitan Adler <lists@eitanadler.com>
sub 4096R/18763D51 2011-02-11
sub 4096R/DAB9CF9B 2011-02-11
```

D.3.4. Shaun Amott <shaun@FreeBSD.org>

```
pub 1024D/6B387A9A 2001-03-19
    Key fingerprint = B506 E6C7 74A1 CC11 9A23 5C13 9268 5D08 6B38 7A9A
uid          Shaun Amott <shaun@inerd.com>
uid          Shaun Amott <shaun@FreeBSD.org>
sub 2048g/26FA8703 2001-03-19
sub 2048R/7FFF5151 2005-11-06
sub 2048R/27C54137 2005-11-06
```

D.3.5. Henrik Brix Andersen <brix@FreeBSD.org>

```
pub 1024D/54E278F8 2003-04-09
    Key fingerprint = 7B63 EF32 7831 A704 220D 7E61 BFE4 387E 54E2 78F8
uid          Henrik Brix Andersen <henrik@brixandersen.dk>
uid          Henrik Brix Andersen <brix@FreeBSD.org>
uid          Henrik Brix Andersen <hbn@terma.com>
uid          Henrik Brix Andersen <brix@osaa.dk>
sub 1024g/3B13C209 2003-04-09
```

D.3.6. Matthias Andree <mandree@FreeBSD.org>

```
pub 1024D/052E7D95 2003-08-28
    Key fingerprint = FDD0 0C43 6E33 07E1 0758 C6A8 BE61 8339 052E 7D95
uid          Matthias Andree <mandree@freebsd.org>
uid          Matthias Andree <matthias.andree@gmx.de>
sub 1536g/E65A83DA 2003-08-28
```

D.3.7. Will Andrews <will@FreeBSD.org>

```
pub 1024D/F81672C5 2000-05-22 Will Andrews (Key for official matters) <will@FreeBSD.org>
    Key fingerprint = 661F BBF7 9F5D 3D02 C862 5F6C 178E E274 F816 72C5
uid          Will Andrews <will@physics.purdue.edu>
uid          Will Andrews <will@puck.firepipe.net>
uid          Will Andrews <will@c-60.org>
uid          Will Andrews <will@csociety.org>
uid          Will Andrews <will@csociety.ecn.purdue.edu>
uid          Will Andrews <will@telperion.openpackages.org>
```

```
sub 1024g/55472804 2000-05-22
```

D.3.8. Dimitry Andric <dim@FreeBSD.org>

```
pub 1024D/2E2096A3 1997-11-17
Key fingerprint = 7AB4 62D2 CE35 FC6D 4239 4FCD B05E A30A 2E20 96A3
uid Dimitry Andric <dimitry@andric.com>
uid Dimitry Andric <dim@xs4all.nl>
uid Dimitry Andric <dimitry.andric@tomtom.com>
uid [jpeg image of size 5132]
uid Dimitry Andric <dim@nah6.com>
uid Dimitry Andric <dim@FreeBSD.org>
sub 4096g/6852A5C5 1997-11-17
```

D.3.9. Eric Anholt <anholt@FreeBSD.org>

```
pub 1024D/6CF0EAF7 2003-09-08
Key fingerprint = 76FE 2475 820B B75F DCA4 0F3E 1D47 6F60 6CF0 EAF7
uid Eric Anholt <eta@lclark.edu>
uid Eric Anholt <anholt@FreeBSD.org>
sub 1024g/80B404C1 2003-09-08
```

D.3.10. Marcus von Appen <mva@FreeBSD.org>

```
pub 1024D/B267A647 2009-02-14
Key fingerprint = C7CC 1853 D8C5 E580 7795 B654 8BAF 3F12 B267 A647
uid Marcus von Appen <freebsd@sysfault.org>
uid Marcus von Appen <mva@freebsd.org>
sub 2048g/D34A3BAF 2009-02-14
```

D.3.11. Marcelo Araujo <araujo@FreeBSD.org>

```
pub 1024D/53E4CFA8 2007-04-27
Key fingerprint = 9D6A 2339 925C 4F61 ED88 ED8B A2FC 4977 53E4 CFA8
uid Marcelo Araujo (Ports Committer) <araujo@FreeBSD.org>
sub 2048g/63CC012D 2007-04-27
```

D.3.12. Mathieu Arnold <mat@FreeBSD.org>

```
pub 1024D/FE6D850F 2005-04-25
Key fingerprint = 2771 11F4 0A7E 73F9 ADDD A542 26A4 7C6A FE6D 850F
uid Mathieu Arnold <mat@FreeBSD.org>
uid Mathieu Arnold <mat@mat.cc>
uid Mathieu Arnold <mat@cpan.org>
```

```
uid          Mathieu Arnold <m@absolight.fr>
uid          Mathieu Arnold <m@absolight.net>
uid          Mathieu Arnold <mat@club-internet.fr>
uid          Mathieu Arnold <marnold@april.org>
uid          Mathieu Arnold <paypal@mat.cc>
sub 2048g/EAD18BD9 2005-04-25
```

D.3.13. Satoshi Asami <asami@FreeBSD.org>

```
pub 1024R/1E08D889 1997-07-23 Satoshi Asami <asami@cs.berkeley.edu>
Key fingerprint = EB 3C 68 9E FB 6C EB 3F DB 2E 0F 10 8F CE 79 CA
uid          Satoshi Asami <asami@FreeBSD.ORG>
```

D.3.14. Gavin Atkinson <gavin@FreeBSD.org>

```
pub 1024D/A093262B 2005-02-18
Key fingerprint = 313A A79F 697D 3A5C 216A EDF5 935D EF44 A093 262B
uid          Gavin Atkinson <gavin@16squared.co.uk>
uid          Gavin Atkinson (FreeBSD key) <gavin@FreeBSD.org>
uid          Gavin Atkinson (Work e-mail) <ga9@york.ac.uk>
uid          Gavin Atkinson <gavin.atkinson@ury.york.ac.uk>
sub 2048g/58F40B3D 2005-02-18
```

D.3.15. Joseph S. Atkinson <jsa@FreeBSD.org>

```
pub 2048R/21AA7B06 2010-07-14
Key fingerprint = 5B38 63B0 9CCA 12BE 3919 9412 CC9D FC84 21AA 7B06
uid          Joseph S. Atkinson <jsa@FreeBSD.org>
uid          Joseph S. Atkinson <jsa.bsd@gmail.com>
uid          Joseph S. Atkinson <jsa@wickedmachine.net>
sub 2048R/5601C3E3 2010-07-14
```

D.3.16. Philippe Audeoud <jadawin@FreeBSD.org>

```
pub 1024D/C835D40E 2005-04-13
Key fingerprint = D090 8C96 3612 15C9 4E3E 7A4A E498 FC2B C835 D40E
uid          Philippe Audeoud <jadawin@tuxaco.net>
uid          Philippe Audeoud <philippe@tuxaco.net>
uid          Philippe Audeoud <philippe.audeoud@sitadelle.com>
uid          Philippe Audeoud <jadawin@freebsd.org>
sub 2048g/EF8EA329 2005-04-13
```

D.3.17. Timur I. Bakeyev <timur@FreeBSD.org>

```
pub 1024D/60BA1F47 2002-04-27
    Key fingerprint = 84BF EAD1 607D 362F 210E 69B3 0BF0 6412 60BA 1F47
uid      Timur I. Bakeyev (BaT) <timur@bat.ru>
uid      Timur I. Bakeyev <timur@gnu.org>
uid      Timur I. Bakeyev (BaT) <bat@cpan.org>
uid      Timur I. Bakeyev (BaT) <timur@FreeBSD.org>
uid      Timur I. Bakeyev (BaT) <timur@gnome.org>
uid      Timur I. Bakeyev <timur@gnome.org>
sub 2048g/8A5B0042 2002-04-27
```

D.3.18. Glen Barber <gjb@FreeBSD.org>

```
pub 2048R/A0B946A3 2010-08-03 [expires: 2017-04-25]
    Key fingerprint = 78B3 42BA 26C7 B2AC 681E A7BE 524F 0C37 A0B9 46A3
uid      Glen Barber <gjb@FreeBSD.org>
uid      Glen Barber <glen.j.barber@gmail.com>
uid      Glen Barber <gjb@glenbarber.us>
sub 2048R/6C0527E5 2010-08-03
```

D.3.19. Nick Barkas <snb@FreeBSD.org>

```
pub 2048R/DDADB9DC 2010-07-27
    Key fingerprint = B678 6ECB 303D F580 A050 098F BDFF 4F3D DDAD B9DC
uid      S. Nicholas Barkas <snb@freebsd.org>
sub 2048R/36E181FB 2010-07-27
sub 2048R/BDA4BED3 2010-07-29
sub 2048R/782A8737 2010-07-29
```

D.3.20. Simon Barner <barner@FreeBSD.org>

```
pub 1024D/EBADA82A 2000-11-10
    Key fingerprint = 67D1 3562 9A2F 3177 E46A 35ED 0A49 FEFD EBAD A82A
uid      Simon Barner <barner@FreeBSD.org>
uid      Simon Barner <barner@in.tum.de>
uid      Simon Barner <barner@informatik.tu-muenchen.de>
uid      Simon Barner <barner@gmx.de>
sub 2048g/F63052DE 2000-11-10
```

D.3.21. Doug Barton <dougb@FreeBSD.org>

```
pub 2048R/1A1ABC84 2010-03-23
    Key fingerprint = E352 0E14 9D05 3533 C33A 67DB 5CC6 86F1 1A1A BC84
uid      Douglas Barton <dougb@dougbarton.us>
uid      Douglas Barton <dougb@FreeBSD.org>
```

```
uid          [jpeg image of size 6140]
sub 3072R/498795B4 2010-03-23
Key fingerprint = C0BE C1E3 8DC8 D7F4 8E6C 732B 0C14 D9CF 4987 95B4
```

D.3.22. Artem Belevich <art@FreeBSD.org>

```
pub 2048R/9ED4C836 2011-03-28
Key fingerprint = 7400 D541 07ED 3DF3 3E97 F2D5 8BDF 101C 9ED4 C836
uid          Artem Belevich <artemb@gmail.com>
uid          Artem Belevich <art@freebsd.org>
sub 2048R/55B0E4EB 2011-03-28
```

D.3.23. Anton Berezin <tobez@FreeBSD.org>

```
pub 1024D/7A7BA3C0 2000-05-25 Anton Berezin <tobez@catpipe.net>
Key fingerprint = CDD8 560C 174B D8E5 0323 83CE 22CA 584C 7A7B A3C0
uid          Anton Berezin <tobez@tobez.org>
uid          Anton Berezin <tobez@FreeBSD.org>
sub 1024g/ADC71E87 2000-05-25
```

D.3.24. Damien Bergamini <damien@FreeBSD.org>

```
pub 2048R/D129F093 2005-03-02
Key fingerprint = D3AB 28C3 1A4A E219 3145 54FE 220A 7486 D129 F093
uid          Damien Bergamini <damien.bergamini@free.fr>
uid          Damien Bergamini <damien@FreeBSD.org>
sub 2048R/9FBA73A4 2005-03-02
```

D.3.25. Tim Bishop <tdb@FreeBSD.org>

```
pub 1024D/5AE7D984 2000-10-07
Key fingerprint = 1453 086E 9376 1A50 ECF6 AE05 7DCE D659 5AE7 D984
uid          Tim Bishop <tim@bishnet.net>
uid          Tim Bishop <T.D.Bishop@kent.ac.uk>
uid          Tim Bishop <tdb@i-scream.org>
uid          Tim Bishop <tdb@FreeBSD.org>
sub 4096g/7F886031 2000-10-07
```

D.3.26. Martin Blapp <mbr@FreeBSD.org>

```
pub 1024D/D300551E 2001-12-20 Martin Blapp <mb@imp.ch>
Key fingerprint = B434 53FC C87C FE7B 0A18 B84C 8686 EF22 D300 551E
sub 1024g/998281C8 2001-12-20
```

D.3.27. Warren Block <wblock@FreeBSD.org>

```
pub 2048R/A1F360A3 2011-09-14
    Key fingerprint = 3A44 4DEC B304 5191 8A41 C317 5117 4BB6 A1F3 60A3
uid Warren Block <wblock@FreeBSD.org>
uid Warren Block <wblock@wonkity.com>
sub 2048R/51F483F3 2011-09-14
```

D.3.28. Vitaly Bogdanov <bvs@FreeBSD.org>

```
pub 1024D/B32017F7 2005-10-02 Vitaly Bogdanov <gad@gad.glazov.net>
    Key fingerprint = 402E B8E4 53CB 22FF BE62 AE35 A0BF B077 B320 17F7
uid Vitaly Bogdanov <bvs@freebsd.org>
sub 1024g/0E88C62E 2005-10-02
```

D.3.29. Roman Bogorodskiy <novel@FreeBSD.org>

```
pub 2048R/08C2226A 2010-12-03
    Key fingerprint = 8BA4 DF2A D14F 99B6 37E0 0070 C96D 5FFE 08C2 226A
uid Roman Bogorodskiy <bogorodskiy@gmail.com>
uid Roman Bogorodskiy <novel@FreeBSD.org>
uid Roman Bogorodskiy <rbogorodskiy@apache.org>
uid Roman Bogorodskiy <rbogorodskiy@gridynamics.com>
sub 2048R/EC4ED237 2010-12-03
```

D.3.30. Renato Botelho <garga@FreeBSD.org>

```
pub 1024D/2244EDA9 2003-12-16 [expires: 2015-10-18]
    Key fingerprint = 4006 C844 BC51 AD75 CE60 6E24 E824 5B89 2244 EDA9
uid Renato Botelho <garga@FreeBSD.org>
uid Renato Botelho <rbgarga@gmail.com>
uid Renato Botelho <garga@freebsdbrasil.com.br>
uid Renato Botelho <renato@galle.com.br>
uid Renato Botelho <freebsd@galle.com.br>
uid Renato Botelho <garga@brainsoft.com.br>
uid Renato Botelho <garga.bsd@gmail.com>
sub 1024g/7B295760 2003-12-16
```

D.3.31. Alexander Botero-Lowry <alexbl@FreeBSD.org>

```
pub 1024D/12A95A7B 2006-09-13
    Key fingerprint = D0C3 47F8 AE87 C829 0613 3586 24DF F52B 12A9 5A7B
uid Alexander Botero-Lowry <alexbl@FreeBSD.org>
sub 2048g/CA287923 2006-09-13
```

D.3.32. Sofian Brabez <sbz@FreeBSD.org>

```
pub 1024D/2487E57E 2011-03-15 [expires: 2016-03-14]
    Key fingerprint = 05BA DC7E F628 DE3F B241 BFBB 7363 51F4 2487 E57E
uid      Sofian Brabez <sbrabez@gmail.com>
uid      Sofian Brabez <sbz@FreeBSD.org>
uid      Sofian Brabez <sbz@6dev.net>
```

D.3.33. Hartmut Brandt <harti@FreeBSD.org>

```
pub 1024D/5920099F 2003-01-29 Hartmut Brandt <brandt@fokus.fraunhofer.de>
    Key fingerprint = F60D 09A0 76B7 31EE 794B BB91 082F 291D 5920 099F
uid      Hartmut Brandt <harti@freebsd.org>
sub 1024g/21D30205 2003-01-29
```

D.3.34. Oliver Braun <obraun@FreeBSD.org>

```
pub 1024D/EF25B1BA 2001-05-06 Oliver Braun <obraun@unsane.org>
    Key fingerprint = 6A3B 042A 732E 17E4 B6E7 3EAF C0B1 6B7D EF25 B1BA
uid      Oliver Braun <obraun@obraun.net>
uid      Oliver Braun <obraun@freebsd.org>
uid      Oliver Braun <obraun@haskell.org>
sub 1024g/09D28582 2001-05-06
```

D.3.35. Max Brazhnikov <makc@FreeBSD.org>

```
pub 1024D/ACB3CD12 2008-08-18
    Key fingerprint = 4BAA 200E 720A 0BD1 7BB0 9DFD FBD9 08C2 ACB3 CD12
uid      Max Brazhnikov <makc@FreeBSD.org>
uid      Max Brazhnikov <makc@issp.ac.ru>
sub 1024g/5FAA4088 2008-08-18
```

D.3.36. Jonathan M. Bresler <jmb@FreeBSD.org>

```
pub 1024R/97E638DD 1996-06-05 Jonathan M. Bresler <jmb@Bresler.org>
    Key fingerprint = 31 57 41 56 06 C1 40 13 C5 1C E3 E5 DC 62 0E FB
uid      Jonathan M. Bresler <jmb@FreeBSD.ORG>
uid      Jonathan M. Bresler
uid      Jonathan M. Bresler <Jonathan.Bresler@USi.net>
uid      Jonathan M. Bresler <jmb@Frb.GOV>
```

D.3.37. Antoine Brodin <antoine@FreeBSD.org>

```
pub 1024D/50CC2671 2008-02-03
    Key fingerprint = F3F7 72F0 9C4C 9E56 4BE9 44EA 1B80 31F3 50CC 2671
uid      Antoine Brodin <antoine@FreeBSD.org>
sub 2048g/6F4AFBE5 2008-02-03
```

D.3.38. Diane Bruce <db@FreeBSD.org>

```
pub 2048R/8E9CAA7B 2012-05-16
    Key fingerprint = 8B08 E022 705D 0083 64C4 5E60 5148 0C74 8E9C AA7B
uid      Diane Bruce <db@db.net>
uid      Diane Bruce <db@FreeBSD.org>
sub 2048R/932E5985 2012-05-16
```

D.3.39. Christian Brueffer <brueffer@FreeBSD.org>

```
pub 1024D/A0ED982D 2002-10-14 Christian Brueffer <chris@unixpages.org>
    Key fingerprint = A5C8 2099 19FF AACA F41B B29B 6C76 178C A0ED 982D
uid      Christian Brueffer <brueffer@hitnet.rwth-aachen.de>
uid      Christian Brueffer <brueffer@FreeBSD.org>
sub 4096g/1DCC100F 2002-10-14
```

D.3.40. Markus Brueffer <markus@FreeBSD.org>

```
pub 1024D/78F8A8D4 2002-10-21
    Key fingerprint = 3F9B EBE8 F290 E5CC 1447 8760 D48D 1072 78F8 A8D4
uid      Markus Brueffer <markus@brueffer.de>
uid      Markus Brueffer <buff@hitnet.rwth-aachen.de>
uid      Markus Brueffer <mbrueffer@mi.rwth-aachen.de>
uid      Markus Brueffer <markus@FreeBSD.org>
sub 4096g/B7E5C7B6 2002-10-21
```

D.3.41. Sean Bruno <sbruno@FreeBSD.org>

```
pub 2048R/9F4CBB4E 2011-10-19
    Key fingerprint = BD78 EB48 FD8C D981 D959 AAE9 BD94 1F06 9F4C BB4E
uid      Sean Bruno (sbruno) <sbruno@freebsd.org>
sub 2048R/3A48F987 2011-10-19
```

D.3.42. Oleg Bulyzhin <oleg@FreeBSD.org>

```
pub 1024D/78CE105F 2004-02-06
    Key fingerprint = 98CC 3E66 26DE 50A8 DBC4 EB27 AF22 DCEF 78CE 105F
uid      Oleg Bulyzhin <oleg@FreeBSD.org>
uid      Oleg Bulyzhin <oleg@rinet.ru>
sub 1024g/F747C159 2004-02-06
```

D.3.43. Michael Bushkov <bushman@FreeBSD.org>

```
pub 1024D/F694C6E4 2007-03-11 [expires: 2008-03-10]
    Key fingerprint = 4278 4392 BF6B 2864 C48E 0FA9 7216 C73C F694 C6E4
uid      Michael Bushkov <bushman@rsu.ru>
uid      Michael Bushkov <bushman@freebsd.org>
sub 2048g/5A783997 2007-03-11 [expires: 2008-03-10]
```

D.3.44. Jayachandran C. <jchandra@FreeBSD.org>

```
pub 1024D/3316E465 2010-05-19
    Key fingerprint = 320B DB08 4FE3 BCFD 60AF E4DB F486 015F 3316 E465
uid      Jayachandran C. <jchandra@freebsd.org>
sub 2048g/1F7755F9 2010-05-19
```

D.3.45. Jesus R. Camou <jcamou@FreeBSD.org>

```
pub 1024D/C2161947 2005-03-01
    Key fingerprint = 274C B265 48EC 42AE A2CA 47D9 7D98 588A C216 1947
uid      Jesus R. Camou <jcamou@FreeBSD.org>
sub 2048g/F8D2A8DF 2005-03-01
```

D.3.46. José Alonso Cárdenas Márquez <acm@FreeBSD.org>

```
pub 1024D/9B21BC19 2006-07-18
    Key fingerprint = 4156 2EAC A11C 9651 713B 3FC1 195F D4A8 9B21 BC19
uid      Jose Alonso Cardenas Marquez <acm@FreeBSD.org>
sub 2048g/ADA16C52 2006-07-18
```

D.3.47. Pietro Cerutti <gahr@FreeBSD.org>

```
pub 1024D/9571F78E 2006-05-17
    Key fingerprint = 1203 92B5 3919 AF84 9B97 28D6 C0C2 6A98 9571 F78E
uid      Pietro Cerutti <gahr@gahr.ch>
uid      Pietro Cerutti (The FreeBSD Project) <gahr@FreeBSD.org>
sub 2048g/F24227D5 2006-05-17 [expires: 2011-05-16]
```

D.3.48. Dmitry Chagin <dchagin@FreeBSD.org>

```
pub 1024D/738EFCED 2009-02-27
    Key fingerprint = 3F3F 8B87 CE09 9E10 3606 6ACA D2DD 936F 738E FCED
uid          Dmitry Chagin <dchagin@freebsd.org>
uid          Dmitry Chagin (dchagin key) <chagin.dmitry@gmail.com>
sub 2048g/6A3FDF9 2009-02-27
```

D.3.49. Hye-Shik Chang <perky@FreeBSD.org>

```
pub 1024D/CFDB4BA4 1999-04-23 Hye-Shik Chang <perky@FreeBSD.org>
    Key fingerprint = 09D9 57D6 58BA 44DD CAEC 71CD 0D65 2C59 CFDB 4BA4
uid          Hye-Shik Chang <hyeshik@gmail.com>
sub 1024g/A94A8ED1 1999-04-23
```

D.3.50. Jonathan Chen <jon@FreeBSD.org>

```
pub 1024D/2539468B 1999-10-11 Jonathan Chen <jon@spock.org>
    Key fingerprint = EE31 CDA1 A105 C8C9 5365 3DB5 C2FC 86AA 2539 468B
uid          Jonathan Chen <jon@freebsd.org>
uid          Jonathan Chen <chenj@rpi.edu>
uid          Jonathan Chen <spock@acm.rpi.edu>
uid          Jonathan Chen <jon@cs.rpi.edu>
sub 3072g/B81EF1DB 1999-10-11
```

D.3.51. Jonathan Anderson <jonathan@FreeBSD.org>

```
pub 1024D/E3BBCA48 2006-06-17
    Key fingerprint = D7C6 9096 874F 707E 48F8 FAB7 22A6 6E53 E3BB CA48
uid          Jonathan Anderson <jonathan@FreeBSD.org>
uid          Jonathan Anderson <jonathan.anderson@ieee.org>
uid          Jonathan Anderson <anderson@engr.mun.ca>
uid          Jonathan Anderson <jonathan.anderson@mun.ca>
sub 2048g/A703650D 2006-06-17
```

D.3.52. Fukang Chen <loader@FreeBSD.org>

```
pub 1024D/40AB1752 2007-08-01 [expires: 2010-07-31]
    Key fingerprint = 98C4 6E6B 1C21 15E4 5042 01FC C7B7 E152 40AB 1752
uid          loader <loader@FreeBSD.org>
sub 4096g/9E53A5C7 2007-08-01 [expires: 2010-07-31]
```

D.3.53. Luoqi Chen <luoqi@FreeBSD.org>

```
pub 1024D/2926F3BE 2002-02-22 Luoqi Chen <luoqi@FreeBSD.org>
    Key fingerprint = B470 A815 5917 D9F4 37F3 CE2A 4D75 3BD1 2926 F3BE
uid                               Luoqi Chen <luoqi@bricore.com>
uid                               Luoqi Chen <lchen@onetta.com>
sub 1024g/5446EB72 2002-02-22
```

D.3.54. Andrey A. Chernov <ache@FreeBSD.org>

```
pub 1024D/964474DD 2006-12-26
    Key fingerprint = 0F63 1B61 D76D AA23 1591 EA09 560E 582B 9644 74DD
uid                               Andrey Chernov <ache@freebsd.org>
uid                               [jpeg image of size 4092]
sub 2048g/08331894 2006-12-26
```

D.3.55. Alexander V. Chernikov <melifaro@FreeBSD.org>

```
pub 1024D/2675AB69 2008-02-17
    Key fingerprint = 00D2 E063 2FB0 2990 C602 50FD C1C2 7889 2675 AB69
uid                               Alexander V. Chernikov <melifaro@yandex-team.ru>
uid                               Alexander V. Chernikov <melifaro@ipfw.ru>
uid                               Alexander V. Chernikov <melifaro@freebsd.org>
sub 4096g/BC64F40C 2008-02-17
```

D.3.56. Sean Chittenden <seanc@FreeBSD.org>

```
pub 1024D/EE278A28 2004-02-08 Sean Chittenden <sean@chittenden.org>
    Key fingerprint = E41F F441 7E91 6CBA 1844 65CF B939 3C78 EE27 8A28
sub 2048g/55321853 2004-02-08
```

D.3.57. Junho CHOI <cjh@FreeBSD.org>

```
pub 1024D/E60260F5 2002-10-14 CHOI Junho (Work) <cjh@wdb.co.kr>
    Key fingerprint = 1369 7374 A45F F41A F3C0 07E3 4A01 C020 E602 60F5
uid                               CHOI Junho (Personal) <cjh@kr.FreeBSD.org>
uid                               CHOI Junho (FreeBSD) <cjh@FreeBSD.org>
sub 1024g/04A4FDD8 2002-10-14
```

D.3.58. Crist J. Clark <cjc@FreeBSD.org>

```
pub 1024D/FE886AD3 2002-01-25 Crist J. Clark <cjclark@jhu.edu>
    Key fingerprint = F04E CCD7 3834 72C2 707F 0A8F 259F 8F4B FE88 6AD3
uid                               Crist J. Clark <cjclark@alum.mit.edu>
```

```
uid                Crist J. Clark <cjc@freebsd.org>
sub 1024g/9B6BAB99 2002-01-25
```

D.3.59. Joe Marcus Clarke <marcus@FreeBSD.org>

```
pub 1024D/FE14CF87 2002-03-04 Joe Marcus Clarke (FreeBSD committer address) <marcus@FreeBSD.org>
   Key fingerprint = CC89 6407 73CC 0286 28E4 AFB9 6F68 8F8A FE14 CF87
uid                Joe Marcus Clarke <marcus@marcuscom.com>
sub 1024g/B9ACE4D2 2002-03-04
```

D.3.60. Nik Clayton <nik@FreeBSD.org>

```
pub 1024D/2C37E375 2000-11-09 Nik Clayton <nik@freebsd.org>
   Key fingerprint = 15B8 3FFC DDB4 34B0 AA5F 94B7 93A8 0764 2C37 E375
uid                Nik Clayton <nik@slashdot.org>
uid                Nik Clayton <nik@crf-consulting.co.uk>
uid                Nik Clayton <nik@ngo.org.uk>
uid                Nik Clayton <nik@bsdi.com>
sub 1024g/769E298A 2000-11-09
```

D.3.61. Benjamin Close <benjsc@FreeBSD.org>

```
pub 1024D/4842B5B4 2002-04-10
   Key fingerprint = F00D C83D 5F7E 5561 DF91 B74D E602 CAA3 4842 B5B4
uid                Benjamin Simon Close <Benjamin.Close@clearchain.com>
uid                Benjamin Simon Close <benjsc@FreeBSD.org>
uid                Benjamin Simon Close <benjsc@clearchain.com>
sub 2048g/3FA8A57E 2002-04-10
```

D.3.62. Tijl Coosemans <tijl@FreeBSD.org>

```
pub 2048D/20A0B62B 2010-07-13
   Key fingerprint = 39AA F580 6B44 5161 9F86 ED49 7E80 92D8 20A0 B62B
uid                Tijl Coosemans <tijl@coosemans.org>
uid                Tijl Coosemans <tijl@freebsd.org>
sub 2048g/7D71BA74 2010-07-13
```

D.3.63. Raphael Kubo da Costa <rakuco@FreeBSD.org>

```
pub 4096R/18DCEED6 2011-10-03
   Key fingerprint = 6911 54FE BA6E 6106 5789 7099 8DD0 7D21 18DC EED6
uid                Raphael Kubo da Costa (Personal key) <rakuco@FreeBSD.org>
```

D.3.64. Bruce Cran <brucec@FreeBSD.org>

```
pub 2048R/6AF6F99E 2010-01-29
    Key fingerprint = 9A3C AE57 2706 B0E3 4B8A 8374 5787 A72B 6AF6 F99E
uid      Bruce Cran <brucec@FreeBSD.org>
uid      Bruce Cran <bruce@cran.org.uk>
sub 2048R/1D665CEE 2010-01-29
```

D.3.65. Frederic Culot <culot@FreeBSD.org>

```
pub 1024D/34876C5B 2006-08-26
    Key fingerprint = 50EE CE94 E43E BA85 CB67 262B B739 1A26 3487 6C5B
uid      Frederic Culot <culot@FreeBSD.org>
uid      Frederic Culot <frederic@culot.org>
sub 2048g/F1EF901F 2006-08-26
```

D.3.66. Aaron Dalton <aaron@FreeBSD.org>

```
pub 1024D/8811D2A4 2006-06-21 [expires: 2011-06-20]
    Key fingerprint = 8DE0 3CBB 3692 992F 53EF ACC7 BE56 0A4D 8811 D2A4
uid      Aaron Dalton <aaron@freebsd.org>
sub 2048g/304EE8E5 2006-06-21 [expires: 2011-06-20]
```

D.3.67. Baptiste Daroussin <bapt@FreeBSD.org>

```
pub 1024D/49A4E84C 2008-11-19
    Key fingerprint = A14B A5FC B860 86DE 73E2 B24C F244 ED31 49A4 E84C
uid      Baptiste Daroussin <bapt@etoilebsd.net>
uid      Baptiste Daroussin <baptiste.daroussin@gmail.com>
uid      Baptiste Daroussin <bapt@FreeBSD.org>
sub 2048g/54AB46B4 2008-11-19
```

D.3.68. Ceri Davies <ceri@FreeBSD.org>

```
pub 1024D/34B7245F 2002-03-08
    Key fingerprint = 9C88 EB05 A908 1058 A4AE 9959 A1C7 DCC1 34B7 245F
uid      Ceri Davies <ceri@submonkey.net>
uid      Ceri Davies <ceri@FreeBSD.org>
uid      Ceri Davies <ceri@opensolaris.org>
sub 1024g/0C482CBC 2002-03-08
```

D.3.69. Brad Davis <brd@FreeBSD.org>

```
pub 1024D/ED0A754D 2005-05-14 [expires: 2014-02-21]
    Key fingerprint = 5DFD D1A6 BEEE A6D4 B3F5 4236 D362 3291 ED0A 754D
uid Brad Davis <sol4k@sol4k.com>
uid Brad Davis <brd@FreeBSD.org>
sub 2048g/1F29D404 2005-05-14 [expires: 2014-02-21]
```

D.3.70. Pawel Jakub Dawidek <pjd@FreeBSD.org>

```
pub 1024D/B1293F34 2004-02-02 Pawel Jakub Dawidek <Pawel@Dawidek.net>
    Key fingerprint = A3A3 5B4D 9CF9 2312 0783 1B1D 168A EF5D B129 3F34
uid Pawel Jakub Dawidek <pjd@FreeBSD.org>
uid Pawel Jakub Dawidek <pjd@FreeBSD.pl>
sub 2048g/3EEC50A7 2004-02-02 [expires: 2006-02-01]
```

D.3.71. Brian S. Dean <bsd@FreeBSD.org>

```
pub 1024D/723BDEE9 2002-01-23 Brian S. Dean <bsd@FreeBSD.org>
    Key fingerprint = EF49 7ABE 47ED 91B3 FC3D 7EA5 4D90 2FF7 723B DEE9
sub 1024g/4B02F876 2002-01-23
```

D.3.72. Vasil Dimov <vd@FreeBSD.org>

```
pub 1024D/F6C1A420 2004-12-08
    Key fingerprint = B1D5 04C6 26CC 0D20 9525 14B8 170E 923F F6C1 A420
uid Vasil Dimov <vd@FreeBSD.org>
uid Vasil Dimov <vd@datamax.bg>
sub 4096g/A0148C94 2004-12-08
```

D.3.73. Roman Divacky <rdivacky@FreeBSD.org>

```
pub 1024D/3DC2044C 2006-11-15
    Key fingerprint = 6B61 25CA 49BC AAC5 21A9 FA7A 2D51 23E8 3DC2 044C
uid Roman Divacky <rdivacky@freebsd.org>
sub 2048g/39BDCE16 2006-11-15
```

D.3.74. Alexey Dokuchaev <danfe@FreeBSD.org>

```
pub 1024D/3C060B44 2004-08-23 Alexey Dokuchaev <danfe@FreeBSD.org>
    Key fingerprint = D970 08A4 922C 8D63 0C19 8D27 F421 76EE 3C06 0B44
sub 1024g/70BAE967 2004-08-23
```

D.3.75. Dima Dorfman <dd@FreeBSD.org>

```

pub 1024D/69FAE582 2001-09-04
    Key fingerprint = B340 8338 7DA3 4D61 7632 098E 0730 055B 69FA E582
uid      Dima Dorfman <dima@trit.org>
uid      Dima Dorfman <dima@unixfreak.org>
uid      Dima Dorfman <dd@freebsd.org>
sub 2048g/65AF3B89 2003-08-19 [expires: 2005-08-18]
sub 2048g/8DB0CF2C 2005-05-29 [expires: 2007-05-29]

```

D.3.76. Olivier Duchateau <olivierd@FreeBSD.org>

```

pub 2048R/22431859 2012-05-28 [expires: 2017-05-27]
    Key fingerprint = C057 112A 4A27 B5F2 CD8F 6C9A FC5A 0167 2243 1859
uid      Olivier Duchateau <duchateau.olivier@gmail.com>
sub 2048R/63A85BDF 2012-05-28 [expires: 2017-05-27]

```

D.3.77. Bruno Ducrot <bruno@FreeBSD.org>

```

pub 1024D/7F463187 2000-12-29
    Key fingerprint = 7B79 E1D6 F5A1 6614 792F D906 899B 4D28 7F46 3187
uid      Ducrot Bruno (Poup Master) <ducrot@poupinou.org>
sub 1024g/40282874 2000-12-29

```

D.3.78. Alex Dupre <ale@FreeBSD.org>

```

pub 1024D/CE5F554D 1999-06-27 Alex Dupre <sysadmin@alexdupre.com>
    Key fingerprint = DE23 02EA 5927 D5A9 D793 2BA2 8115 E9D8 CE5F 554D
uid      Alex Dupre <ale@FreeBSD.org>
uid      [jpeg image of size 5544]
uid      Alex Dupre <ICQ:5431856>
sub 2048g/FD5E2D21 1999-06-27

```

D.3.79. Peter Edwards <peadar@FreeBSD.org>

```

pub 1024D/D80B4B3F 2004-03-01 Peter Edwards <peadar@FreeBSD.org>
    Key fingerprint = 7A8A 9756 903E BEF2 4D9E 3C94 EE52 52F7 D80B 4B3F
uid      Peter Edwards <pmedwards@eircom.net>

```

D.3.80. Josef El-Rayes <josef@FreeBSD.org>

```

pub 2048R/A79DB53C 2004-01-04 Josef El-Rayes <josef@FreeBSD.org>
    Key fingerprint = 58EB F5B7 2AB9 37FE 33C8 716B 59C5 22D9 A79D B53C
uid      Josef El-Rayes <josef@daemon.li>

```

D.3.81. Lars Engels <lme@FreeBSD.org>

```
pub 1024D/C0F769F8 2004-08-27
    Key fingerprint = 17FC 08E1 5E09 BD21 489E 2050 29CE 75DA C0F7 69F8
uid          Lars Engels <lars.engels@0x20.net>
sub 1024g/8AD5BF9D 2004-08-27
```

D.3.82. Udo Erdelhoff <ue@FreeBSD.org>

```
pub 1024R/E74FA871 1994-07-19 Udo Erdelhoff <uer@de.uu.net>
    Key fingerprint = 8C B1 80 CA 2C 52 73 81 FB A7 B4 03 C5 32 C8 67
uid          Udo Erdelhoff <ue@nathan.ruhr.de>
uid          Udo Erdelhoff <ue@freebsd.org>
uid          Udo Erdelhoff <uerdelho@eu.uu.net>
uid          Udo Erdelhoff <uerdelho@uu.net>
```

D.3.83. Ruslan Ermilov <ru@FreeBSD.org>

```
pub 1024D/996E145E 2004-06-02 Ruslan Ermilov (FreeBSD) <ru@FreeBSD.org>
    Key fingerprint = 274E D201 71ED 11F6 9CCB 0194 A917 E9CC 996E 145E
uid          Ruslan Ermilov (FreeBSD Ukraine) <ru@FreeBSD.org.ua>
uid          Ruslan Ermilov (IPNet) <ru@ip.net.ua>
sub 1024g/557E3390 2004-06-02 [expires: 2007-06-02]
```

D.3.84. Lukas Ertl <le@FreeBSD.org>

```
pub 1024D/F10D06CB 2000-11-23 Lukas Ertl <le@FreeBSD.org>
    Key fingerprint = 20CD C5B3 3A1D 974E 065A B524 5588 79A9 F10D 06CB
uid          Lukas Ertl <a9404849@unet.univie.ac.at>
uid          Lukas Ertl <l.ertl@univie.ac.at>
uid          Lukas Ertl <le@univie.ac.at>
sub 1024g/5960CE8E 2000-11-23
```

D.3.85. Brendan Fabeny <bf@FreeBSD.org>

```
pub 2048R/9806EBC1 2010-06-08 [expires: 2012-06-07]
    Key fingerprint = 2075 ADD3 7634 A4F9 5357 D934 08E7 06D9 9806 EBC1
uid          b. f. <bf@freebsd.org>
sub 2048R/1CD0AD79 2010-06-08 [expires: 2012-06-07]
```

D.3.86. Guido Falsi <madpilot@FreeBSD.org>

```
pub 2048R/56CBD293 2012-04-12
    Key fingerprint = F317 2057 E17E 4E3A 3DA5 9E1D 1AE6 860E 56CB D293
uid Guido Falsi <madpilot@FreeBSD.org>
uid Guido Falsi <mad@madpilot.net>
sub 2048R/1F9772C5 2012-04-12
```

D.3.87. Rong-En Fan <rafan@FreeBSD.org>

```
pub 1024D/86FD8C68 2004-06-04
    Key fingerprint = DC9E 5B4D 2DDA D5C7 B6F8 6E69 D78E 1091 86FD 8C68
uid Rong-En Fan <rafan@infor.org>
uid Rong-En Fan <rafan@csie.org>
uid Rong-En Fan <rafan@FreeBSD.org>
sub 2048g/42A8637E 2009-01-25 [expires: 2012-07-08]
```

D.3.88. Stefan Farfeleder <stefanf@FreeBSD.org>

```
pub 1024D/8BEFD15F 2004-03-14 Stefan Farfeleder <stefan@fafoe.narf.at>
    Key fingerprint = 4220 FE60 A4A1 A490 5213 27A6 319F 8B28 8BEF D15F
uid Stefan Farfeleder <stefanf@complang.tuwien.ac.at>
uid Stefan Farfeleder <stefanf@FreeBSD.org>
uid Stefan Farfeleder <stefanf@ten15.org>
sub 2048g/418753E9 2004-03-14 [expires: 2007-03-14]
```

D.3.89. Babak Farrokhi <farrokhi@FreeBSD.org>

```
pub 1024D/7C810476 2005-12-22
    Key fingerprint = AABD 388F A207 58B4 2EE3 5DFD 4FC1 32C3 7C81 0476
uid Babak Farrokhi <farrokhi@FreeBSD.org>
uid Babak Farrokhi <babak@farrokhi.net>
sub 2048g/2A5F93C7 2005-12-22
```

D.3.90. Chris D. Faulhaber <jedgar@FreeBSD.org>

```
pub 1024D/FE817A50 2000-12-20 Chris D. Faulhaber <jedgar@FreeBSD.org>
    Key fingerprint = A47D A838 9216 F921 A456 54FF 39B6 86E0 FE81 7A50
uid Chris D. Faulhaber <jedgar@fxp.org>
sub 2048g/93452698 2000-12-20
```

D.3.91. Brian F. Feldman <green@FreeBSD.org>

```
pub 1024D/41C13DE3 2000-01-11 Brian Fundakowski Feldman <green@FreeBSD.org>
   Key fingerprint = 6A32 733A 1BF6 E07B 5B8D AE14 CC9D DCA2 41C1 3DE3
sub 1024g/A98B9FCC 2000-01-11 [expires: 2001-01-10]

pub 1024D/773905D6 2000-09-02 Brian Fundakowski Feldman <green@FreeBSD.org>
   Key fingerprint = FE23 7481 91EA 5E58 45EA 6A01 B552 B043 7739 05D6
sub 2048g/D2009B98 2000-09-02
```

D.3.92. Mário Sérgio Fujikawa Ferreira <lioux@FreeBSD.org>

```
pub 1024D/75A63712 2006-02-23 [expires: 2007-02-23]
   Key fingerprint = 42F2 2F74 8EF9 5296 898F C981 E9CF 463B 75A6 3712
uid          Mario Sergio Fujikawa Ferreira (lioux) <lioux@FreeBSD.org>
uid          Mario Sergio Fujikawa Ferreira <lioux@uol.com.br>
sub 4096g/BB7D80F2 2006-02-23 [expires: 2007-02-23]
```

D.3.93. Tony Finch <fanf@FreeBSD.org>

```
pub 1024D/84C71B6E 2002-05-03 Tony Finch <dot@dotat.at>
   Key fingerprint = 199C F25B 2679 6D04 63C5 2159 FFC0 F14C 84C7 1B6E
uid          Tony Finch <fanf@FreeBSD.org>
uid          Tony Finch <fanf@apache.org>
uid          Tony Finch <fanf2@cam.ac.uk>
sub 2048g/FD101E8B 2002-05-03
```

D.3.94. Marc Fonvieille <blackend@FreeBSD.org>

```
pub 1024D/4F8E74E8 2004-12-25 Marc Fonvieille <blackend@FreeBSD.org>
   Key fingerprint = 55D3 4883 4A04 828A A139 A5CF CD0F 51C0 4F8E 74E8
uid          Marc Fonvieille <marc@blackend.org>
uid          Marc Fonvieille <marc@freebsd-fr.org>
sub 1024g/37AD4E7D 2004-12-25
```

D.3.95. Pete Fritchman <petef@FreeBSD.org>

```
pub 1024D/74B91CFD 2001-01-30 Pete Fritchman <petef@FreeBSD.org>
   Key fingerprint = 9A9F 8A13 DB0D 7777 8D8E 1CB2 C5C9 A08F 74B9 1CFD
uid          Pete Fritchman <petef@databits.net>
uid          Pete Fritchman <petef@csh.rit.edu>
sub 1024g/0C02AF0C 2001-01-30
```

D.3.96. Bernhard Fröhlich <decke@FreeBSD.org>

```
pub 1024D/CF5840D4 2008-01-07 [expires: 2015-05-05]
    Key fingerprint = 47F6 BDF1 DF9E 81E2 2C54 8A06 E796 7A5A CF58 40D4
uid      Bernhard Fröhlich <decke@FreeBSD.org>
uid      Bernhard Fröhlich <decke@bluelife.at>
sub 2048g/4E51CE79 2008-01-07
```

D.3.97. Bill Fumerola <billf@FreeBSD.org>

```
pub 1024D/7F868268 2000-12-07 Bill Fumerola (FreeBSD Developer) <billf@FreeBSD.org>
    Key fingerprint = 5B2D 908E 4C2B F253 DAEB FC01 8436 B70B 7F86 8268
uid      Bill Fumerola (Security Yahoo) <fumerola@yahoo-inc.com>
sub 1024g/43980DA9 2000-12-07
```

D.3.98. Andriy Gapon <avg@FreeBSD.org>

```
pub 2048R/A651FE2F 2009-02-16
    Key fingerprint = F234 4D58 DEFF 5E3A 4E0F 13BC 74A5 2D27 A651 FE2F
uid      Andriy Gapon (FreeBSD) <avg@FreeBSD.org>
uid      Andriy Gapon (FreeBSD) <avg@freebsd.org>
uid      Andriy Gapon (FreeBSD) <avg@icyb.net.ua>
sub 4096R/F9A4D312 2009-02-16
```

D.3.99. Beat Gätzi <beat@FreeBSD.org>

```
pub 1024D/774249DB 2009-01-28 [expires: 2014-01-27]
    Key fingerprint = C410 3187 5B29 DD02 745F 0890 40C5 BCF7 7742 49DB
uid      Beat Gaetzi <beat@FreeBSD.org>
sub 2048g/173CFFCA 2009-01-28 [expires: 2014-01-27]
```

D.3.100. Daniel Geržo <danger@FreeBSD.org>

```
pub 1024D/DA913352 2007-08-30 [expires: 2008-08-29]
    Key fingerprint = 7372 3F15 F839 AFF5 4052 CAC7 1ADA C204 DA91 3352
uid      Daniel Gerzo <gerzo@rulez.sk>
uid      Daniel Gerzo <danger@rulez.sk>
uid      Daniel Gerzo (The FreeBSD Project) <danger@FreeBSD.org>
uid      Daniel Gerzo (Micronet, a.s.) <gerzo@micronet.sk>
sub 2048g/C5D57BDC 2007-08-30 [expires: 2008-08-29]
```

D.3.101. Justin T. Gibbs <gibbs@FreeBSD.org>

```
pub 2048R/45A4FC2F 2012-02-10
    Key fingerprint = B98A C3AB 412B 094B D6FE E713 FA5A 1E30 45A4 FC2F
uid Justin T. Gibbs <gibbs@FreeBSD.org>
uid Justin T. Gibbs <gibbs@FreeBSDFoundation.org>
uid Justin T. Gibbs <gibbs@scsiguy.com>
sub 2048R/AF6927F8 2012-02-10
```

D.3.102. Pedro Giffuni <pfg@FreeBSD.org>

```
pub 2048D/422BDFE4 2011-12-06
    Key fingerprint = A12B 7C6B 54C0 921B C64F 7B35 58DF 6813 422B DFE4
uid Pedro Giffuni (FreeBSD key signature) <pfg@FreeBSD.org>
sub 2048g/43A91DE0 2011-12-06
```

D.3.103. Philip M. Gollucci <pgollucci@FreeBSD.org>

```
pub 1024D/DB9B8C1C 2008-04-15
    Key fingerprint = B90B FBC3 A3A1 C71A 8E70 3F8C 75B8 8FFB DB9B 8C1C
uid Philip M. Gollucci (FreeBSD Foundation) <pgollucci@freebsd.org>
uid Philip M. Gollucci (Riderway Inc.) <pgollucci@riderway.com>
uid Philip M. Gollucci <pgollucci@p6m7g8.com>
uid Philip M. Gollucci (ASF) <pgollucci@apache.org>
sub 2048g/73943732 2008-04-15
```

D.3.104. Daichi GOTO <daichi@FreeBSD.org>

```
pub 1024D/09EBADD6 2002-09-25 Daichi GOTO <daichi@freebsd.org>
    Key fingerprint = 620A 9A34 57FB 5E93 0828 28C7 C360 C6ED 09EB ADD6
sub 1024g/F0B1F1CA 2002-09-25
```

D.3.105. Marcus Alves Grando <mnag@FreeBSD.org>

```
pub 1024D/CDCC273F 2005-09-15 [expires: 2010-09-14]
    Key fingerprint = 57F9 DEC1 5BBF 06DE 44A5 9A4A 8BEE 5F3A CDCC 273F
uid Marcus Alves Grando <marcus@sbh.eng.br>
uid Marcus Alves Grando <marcus@corp.grupos.com.br>
uid Marcus Alves Grando <mnag@FreeBSD.org>
sub 2048g/698AC00C 2005-09-15 [expires: 2010-09-14]
```

D.3.106. Peter Grehan <grehan@FreeBSD.org>

```
pub 1024D/EA45EA7D 2004-07-13 Peter Grehan <grehan@freebsd.org>
   Key fingerprint = 84AD 73DC 370E 15CA 7556 43C8 F5C8 4450 EA45 EA7D
sub 2048g/0E122D70 2004-07-13
```

D.3.107. Jamie Gritton <jamie@FreeBSD.org>

```
pub 1024D/8832CB7F 2009-01-29
   Key fingerprint = 34F8 1E62 C7A5 7CB9 A91F 7864 8C5A F85E 8832 CB7F
uid          James Gritton <jamie@FreeBSD.org>
sub 2048g/94E3594D 2009-01-29
```

D.3.108. John-Mark Gurney <jmg@FreeBSD.org>

```
pub 1024R/3F9951F5 1997-02-11 John-Mark Gurney <johnmark@gladstone.uoregon.edu>
   Key fingerprint = B7 EC EF F8 AE ED A7 31 96 7A 22 B3 D8 56 36 F4
uid          John-Mark Gurney <gurney_j@efn.org>
uid          John-Mark Gurney <jmg@cs.uoregon.edu>
uid          John-Mark Gurney <gurney_j@resnet.uoregon.edu>
```

D.3.109. Mateusz Guzik <mjg@FreeBSD.org>

```
pub 2048R/21489259 2012-06-03
   Key fingerprint = 3A9F 25FF ABF6 BB23 5C70 C61B 96D3 5178 2148 9259
uid          Mateusz Guzik <mjg@freebsd.org>
sub 2048R/EA19FE8D 2012-06-03
```

D.3.110. Daniel Harris <dannyboy@FreeBSD.org>

```
pub 1024D/84D0D7E7 2001-01-15 Daniel Harris <dannyboy@worksforfood.com>
   Key fingerprint = 3C61 B8A1 3F09 D194 3259 7173 6C63 DA04 84D0 D7E7
uid          Daniel Harris <dannyboy@freebsd.org>
uid          Daniel Harris <dh@askdh.com>
uid          Daniel Harris <dh@wordassault.com>
sub 1024g/9DF0231A 2001-01-15
```

D.3.111. Daniel Hartmeier <dhartmei@FreeBSD.org>

```
pub 1024R/6A3A7409 1994-08-15 Daniel Hartmeier <dhartmei@freebsd.org>
   Key fingerprint = 13 7E 9A F3 36 82 09 FE FD 57 B8 5C 2B 81 7E 1F
```

D.3.112. Olli Hauer <ohauer@FreeBSD.org>

```
pub 2048R/5D008F1A 2010-07-26
    Key fingerprint = E9EE C9A5 EB4C BD29 74D7 9178 E56E 06B3 5D00 8F1A
uid olli hauer <ohauer@FreeBSD.org>
uid olli hauer <ohauer@gmx.de>
sub 2048R/5E25776E 2010-07-26
```

D.3.113. Emanuel Haupt <ehaupt@FreeBSD.org>

```
pub 2048R/C06D09BE 2010-09-24 [expires: 2011-09-24]
    Key fingerprint = CC88 5081 78D1 39C3 B467 865A 348E F6CC C06D 09BE
uid Emanuel Haupt <ehaupt@FreeBSD.org>
sub 2048R/F658659F 2010-09-24 [expires: 2011-09-24]
```

D.3.114. John Hay <jhay@FreeBSD.org>

```
pub 2048R/A9275B93 2000-05-10 John Hay <jhay@icomtek.csir.co.za>
    Key fingerprint = E7 95 F4 B9 D4 A7 49 6A 83 B9 77 49 28 9E 37 70
uid John Hay <jhay@mikom.csir.co.za>
uid Thawte Freemail Member <jhay@mikom.csir.co.za>
uid John Hay <jhay@csir.co.za>
uid John Hay <jhay@FreeBSD.ORG>
```

D.3.115. Sheldon Hearn <sheldonh@FreeBSD.org>

```
pub 1024D/74A06ACD 2002-06-20 Sheldon Hearn <sheldonh@starjuice.net>
    Key fingerprint = 01A3 EF91 9C5A 3633 4E01 8085 A462 57F1 74A0 6ACD
sub 1536g/C42F8AC8 2002-06-20
```

D.3.116. Mike Heffner <mikeh@FreeBSD.org>

```
pub 1024D/CDECBF99 2001-02-02 Michael Heffner <mheffner@novacoxmail.com>
    Key fingerprint = AFAB CCEB 68C7 573F 5110 9285 1689 1942 CDEC BF99
uid Michael Heffner <mheffner@vt.edu>
uid Michael Heffner <mikeh@FreeBSD.org>
uid Michael Heffner <spock@techfour.net>
uid Michael Heffner (ACM sysadmin) <mheffner@acm.vt.edu>
sub 1024g/3FE83FB5 2001-02-02
```

D.3.117. Martin Heinen <mheinen@FreeBSD.org>

```
pub 1024D/116C5C85 2002-06-17 Martin Heinen <mheinen@freebsd.org>
    Key fingerprint = C898 3FCD EEA0 17ED BEA9 564D E5A6 AFF2 116C 5C85
uid                                     Martin Heinen <martin@sumuk.de>
sub 1024g/EA67506B 2002-06-17
```

D.3.118. Niels Heinen <niels@FreeBSD.org>

```
pub 1024D/5FE39B80 2004-12-06 Niels Heinen <niels.heinen@ubizen.com>
    Key fingerprint = 75D8 4100 CF5B 3280 543F 930C 613E 71AA 5FE3 9B80
uid                                     Niels Heinen <niels@defaced.be>
uid                                     Niels Heinen <niels@heinen.ws>
uid                                     Niels Heinen <niels@FreeBSD.org>
sub 2048g/057F4DA7 2004-12-06
```

D.3.119. Jaakko Heinonen <jh@FreeBSD.org>

```
pub 1024D/53CCB781 2009-10-01 [expires: 2014-09-30]
    Key fingerprint = 3AED A2B6 B63D D771 1AFD 25FA DFDF 5B89 53CC B781
uid                                     Jaakko Heinonen (FreeBSD) <jh@FreeBSD.org>
sub 4096g/BB97397E 2009-10-01 [expires: 2014-09-30]
```

D.3.120. Jason Helfman <jgh@FreeBSD.org>

```
pub 2048R/4150D3DC 2011-12-18 [expires: 2021-12-15]
    Key fingerprint = 8E0D C457 9A0F C91C 23F3 0454 2059 9A63 4150 D3DC
uid                                     Jason Helfman <jgh@FreeBSD.org>
sub 2048R/695B1B92 2011-12-18 [expires: 2021-12-15]
```

D.3.121. Guy Helmer <ghelmer@FreeBSD.org>

```
pub 2048R/8F1CEBC4 2012-05-22
    Key fingerprint = 483E 9E6C C644 2520 C9FE 4E87 9989 CCAF 8F1C EBC4
uid                                     Guy Helmer <guy.helmer@palisadesystems.com>
uid                                     Guy Helmer <guy.helmer@gmail.com>
uid                                     Guy Helmer <ghelmer@freebsd.org>
sub 2048R/2073E3F8 2012-05-22

pub 1024R/35F4ED2D 1997-01-26 Guy G. Helmer <ghelmer@freebsd.org>
    Key fingerprint = A2 59 4B 92 02 5B 9E B1 B9 4E 2E 03 29 D5 DC 3A
uid                                     Guy G. Helmer <ghelmer@cs.iastate.edu>
uid                                     Guy G. Helmer <ghelmer@palisadesys.com>
```

D.3.122. Maxime Henrion <mux@FreeBSD.org>

```
pub 1024D/881D4806 2003-01-09 Maxime Henrion <mux@FreeBSD.org>
    Key fingerprint = 81F1 BE2D 12F1 184A 77E4 ACD0 5563 7614 881D 4806
sub 2048g/D0B510C0 2003-01-09
```

D.3.123. Dennis Herrmann <dhn@FreeBSD.org>

```
pub 1024D/65181EA0 2008-09-07 [expires: 2009-03-06]
    Key fingerprint = D4DB A438 EB5E 1B26 C782 F969 820B 66B3 6518 1EA0
uid Dennis Herrmann (Vi veri universum vivus vici) <adox@mcx2.org>
sub 4096g/C003C5DD 2008-09-07 [expires: 2009-03-06]
```

D.3.124. Justin Hibbits <jhibbits@FreeBSD.org>

```
pub 2048R/37BE2DB9 2011-12-01
    Key fingerprint = 8A12 7064 4F3D 339A 191D AD52 30C7 858E 37BE 2DB9
uid Justin Hibbits <chmeedalf@gmail.com>
uid Justin Hibbits <jhibbits@freebsd.org>
uid Justin Hibbits <jrh29@alumni.cwru.edu>
sub 2048R/A8DA156F 2011-12-01
```

D.3.125. Peter Holm <pho@FreeBSD.org>

```
pub 1024D/CF244E81 2008-11-17
    Key fingerprint = BE9B 32D8 89F1 F285 00E4 E4C5 EF3F B4B5 CF24 4E81
uid Peter Holm <pho@FreeBSD.org>
sub 2048g/E20A409F 2008-11-17
```

D.3.126. Michael L. Hostbaek <mich@FreeBSD.org>

```
pub 1024D/0F55F6BE 2001-08-07 Michael L. Hostbaek <mich@freebsdcluster.org>
    Key fingerprint = 4D62 9396 B19F 38D3 5C99 1663 7B0A 5212 0F55 F6BE
uid Michael L. Hostbaek <mich@freebsdcluster.dk>
uid Michael L. Hostbaek <mich@icommerce-france.com>
uid Micahel L. Hostbaek <mich@freebsd.dk>
uid Michael L. Hostbaek <mich@the-lab.org>
uid Michael L. Hostbaek <mich@freebsd.org>
sub 1024g/8BE4E30F 2001-08-07
```

D.3.127. Po-Chuan Hsieh <sunpoet@FreeBSD.org>

```
pub 4096R/CC57E36B 2010-09-21
    Key fingerprint = 8AD8 68F2 7D2B 0A10 7E9B 8CC0 DC44 247E CC57 E36B
uid Po-Chuan Hsieh (FreeBSD) <sunpoet@FreeBSD.org>
uid Po-Chuan Hsieh (sunpoet) <sunpoet@sunpoet.net>
sub 4096R/ADE9E203 2010-09-21
```

D.3.128. Li-Wen Hsu <lwhsu@FreeBSD.org>

```
pub 1024D/2897B228 2005-01-16
    Key fingerprint = B6F7 170A 6DC6 5D1A BD4B D86A 416B 0E39 2897 B228
uid Li-wen Hsu <lwhsu@lwhsu.org>
uid Li-wen Hsu <lwhsu@lwhsu.ckefgisc.org>
uid Li-wen Hsu <lwhsu@lwhsu.csie.net>
uid Li-wen Hsu <lwhsu@ckefgisc.org>
uid Li-wen Hsu <lwhsu@csie.nctu.edu.tw>
uid Li-wen Hsu <lwhsu@ccca.nctu.edu.tw>
uid Li-wen Hsu <lwhsu@iis.sinica.edu.tw>
uid Li-wen Hsu <lwhsu@cs.nctu.edu.tw>
uid Li-Wen Hsu <lwhsu@FreeBSD.org>
sub 2048g/16F82238 2005-01-16
```

D.3.129. Howard F. Hu <foxfair@FreeBSD.org>

```
pub 1024D/4E9BCA59 2003-09-01 Foxfair Hu <foxfair@FreeBSD.org>
    Key fingerprint = 280C A846 CA1B CAC9 DDCF F4CB D553 4BD5 4E9B CA59
uid Foxfair Hu <foxfair@drago.fomokka.net>
uid Howard Hu <howardhu@yahoo-inc.com>
sub 1024g/3356D8C1 2003-09-01
```

D.3.130. Chin-San Huang <chinsan@FreeBSD.org>

```
pub 1024D/350EECF8 2006-10-04
    Key fingerprint = 1C4D 0C9E 0E68 DB74 0688 CE43 D2A5 3F82 350E ECFA
uid Chin-San Huang (lab) <chinsan@chinsan2.twbbs.org>
uid Chin-San Huang (FreeBSD committer) <chinsan@FreeBSD.org>
uid Chin-San Huang (Gmail) <chinsan.tw@gmail.com>
sub 2048g/35F75A30 2006-10-04
```

D.3.131. Davide Italiano <davide@FreeBSD.org>

```
pub 2048R/4CB47484 2012-01-17
    Key fingerprint = B5C9 77F5 1E67 D110 8D19 7587 EB95 EA82 4CB4 7484
uid Davide Italiano <davide@FreeBSD.org>
sub 2048R/91F7443D 2012-01-17
```

D.3.132. Jordan K. Hubbard <jkh@FreeBSD.org>

```
pub 1024R/8E542D5D 1996-04-04 Jordan K. Hubbard <jkh@FreeBSD.org>
Key fingerprint = 3C F2 27 7E 4A 6C 09 0A 4B C9 47 CD 4F 4D 0B 20
```

D.3.133. Konrad Jankowski <versus@FreeBSD.org>

```
pub 1024D/A01C218A 2008-10-28
Key fingerprint = A805 21DC 859F E941 D2EA 9986 2264 8E5D A01C 218A
uid Konrad Jankowski <versus@freebsd.org>
sub 2048g/56AE1959 2008-10-28
```

D.3.134. Weongyo Jeong <weongyo@FreeBSD.org>

```
pub 1024D/22354D7A 2007-12-28
Key fingerprint = 138E 7115 A86F AA40 B509 5883 B387 DCE9 2235 4D7A
uid Weongyo Jeong <weongyo.jeong@gmail.com>
uid Weongyo Jeong <weongyo@freebsd.org>
sub 2048g/9AE6DAEE 2007-12-28
```

D.3.135. Tatuja JINMEI <jinmei@FreeBSD.org>

```
pub 1024D/ABA82228 2002-08-15
Key fingerprint = BB70 3050 EE39 BE00 48BB A5F3 5892 F203 ABA8 2228
uid JINMEI Tatuja <jinmei@FreeBSD.org>
uid JINMEI Tatuja <jinmei@jinmei.org>
uid JINMEI Tatuja (the KAME project) <jinmei@isl.rdc.toshiba.co.jp>
sub 1024g/8B43CF66 2002-08-15
```

D.3.136. Michael Johnson <ahze@FreeBSD.org>

```
pub 1024D/3C046FD6 2004-10-29 Michael Johnson (FreeBSD key) <ahze@FreeBSD.org>
Key fingerprint = 363C 6ABA ED24 C23B 5F0C 3AB4 9F8B AA7D 3C04 6FD6
uid Michael Johnson (pgp key) <ahze@ahze.net>
sub 2048g/FA334AE3 2004-10-29
```

D.3.137. Trevor Johnson <trevor@FreeBSD.org>

```
pub 1024D/3A3EA137 2000-04-20 Trevor Johnson <trevor@jpj.net>
Key fingerprint = 7ED1 5A92 76C1 FFCB E5E3 A998 F037 5A0B 3A3E A137
sub 1024g/46C24F1E 2000-04-20
```

D.3.138. Tom Judge <tj@FreeBSD.org>

```
pub 2048R/81E22216 2012-05-27 [expires: 2017-05-26]
    Key fingerprint = 8EF8 36C8 44A6 9576 6ADB EB0E 4252 33DC 81E2 2216
uid Tom Judge <tom@tomjudge.com>
uid Tom Judge <tjudge@sourcefire.com>
uid Tom Judge <tj@freebsd.org>
sub 2048R/2CA4AA0D 2012-05-27 [expires: 2017-05-26]
```

D.3.139. Benjamin Kaduk <bjk@FreeBSD.org>

```
pub 4096R/8302FE9F 2011-08-20 [expires: 2013-07-21]
    Key fingerprint = 9FD9 F966 D914 5101 BE59 FE13 2D29 EEED 8302 FE9F
uid Benjamin Kaduk <bjk@FreeBSD.org>
sub 4096R/28698ABE 2011-08-20 [expires: 2013-08-19]
```

D.3.140. Poul-Henning Kamp <phk@FreeBSD.org>

```
pub 1024R/0358FCBD 1995-08-01 Poul-Henning Kamp <phk@FreeBSD.org>
    Key fingerprint = A3 F3 88 28 2F 9B 99 A2 49 F4 E2 FA 5A 78 8B 3E
```

D.3.141. Sergey Kandaurov <pluknet@FreeBSD.org>

```
pub 2048R/10607419 2010-10-04
    Key fingerprint = 020B EC25 7E1F 8BC5 C42C 513B 3F4E 97BA 1060 7419
uid Sergey Kandaurov (freebsd) <pluknet@freebsd.org>
uid Sergey Kandaurov <pluknet@gmail.com>
sub 2048R/5711F73B 2010-10-04
```

D.3.142. Coleman Kane <cokane@FreeBSD.org>

```
pub 1024D/C5DAB797 2007-07-22
    Key fingerprint = FC09 F326 4318 E714 DE45 6CB0 70C4 B141 C5DA B797
uid Coleman Kane (Personal PGP Key) <cokane@cokane.org>
uid Coleman Kane (Personal PGP Key) <cokane@FreeBSD.org>
sub 2048g/5C680129 2007-07-22
```

D.3.143. Josef Karthausser <joe@FreeBSD.org>

```
pub 1024D/E6B15016 2000-10-19 Josef Karthausser <joe@FreeBSD.org>
    Key fingerprint = 7266 8EAF 82C2 D439 5642 AC26 5D52 1C8C E6B1 5016
uid Josef Karthausser <joe@tao.org.uk>
uid Josef Karthausser <joe@uk.FreeBSD.org>
uid [revoked] Josef Karthausser <josef@bsd.i.com>
```

```
uid [revoked] Josef Karthauser <joe@pavilion.net>
sub 2048g/1178B692 2000-10-19
```

D.3.144. Vinod Kashyap <vkashyap@FreeBSD.org>

```
pub 1024R/04FCCDD3 2004-02-19 Vinod Kashyap (gnupg key) <vkashyap@freebsd.org>
Key fingerprint = 9B83 0B55 604F E491 B7D2 759D DF92 DAA0 04FC CDD3
```

D.3.145. Kris Kennaway <kris@FreeBSD.org>

```
pub 1024D/68E840A5 2000-01-14 Kris Kennaway <kris@citusc.usc.edu>
Key fingerprint = E65D 0E7D 7E16 B212 1BD6 39EE 5ABC B405 68E8 40A5
uid Kris Kennaway <kris@FreeBSD.org>
uid Kris Kennaway <kris@obsecrity.org>
sub 2048g/03A41C45 2000-01-14 [expires: 2006-01-14]
```

D.3.146. Giorgos Keramidas <keramida@FreeBSD.org>

```
pub 1024D/318603B6 2001-09-21
Key fingerprint = C1EB 0653 DB8B A557 3829 00F9 D60F 941A 3186 03B6
uid Giorgos Keramidas <keramida@FreeBSD.org>
uid Giorgos Keramidas <keramida@ceid.upatras.gr>
uid Giorgos Keramidas <keramida@hellug.gr>
uid Giorgos Keramidas <keramida@linux.gr>
uid Giorgos Keramidas <gkeramidas@gmail.com>
sub 1024g/50FDBAD1 2001-09-21
```

D.3.147. Max Khon <fjoe@FreeBSD.org>

```
pub 1024D/6B87E212 2009-02-17
Key fingerprint = 124D EC6C 6365 D41A 497A 9C3E FCF3 8708 6B87 E212
uid Max Khon <fjoe@FreeBSD.org>
uid Max Khon <fjoe@samodelkin.net>
sub 2048g/CB71491D 2009-02-17
```

D.3.148. Manolis Kiagias <manolis@FreeBSD.org>

```
pub 1024D/6E0FB494 2006-08-22
Key fingerprint = F820 5AAF 7112 2CDD 23D8 3BDF 67F3 311A 6E0F B494
uid Manolis Kiagias <manolis@FreeBSD.org>
uid Manolis Kiagias <sonicy@otenet.gr>
uid Manolis Kiagias (A.K.A. sonic, sonicy, sonic2000gr) <sonic@diktia.dyndns.org>
sub 2048g/EB94B411 2006-08-22
```

D.3.149. Jung-uk Kim <jkim@FreeBSD.org>

```
pub 1024D/BF6A9D53 2004-04-07
    Key fingerprint = F841 0339 93EF D27D 32AD 3261 9A56 B2D5 BF6A 9D53
uid Jung-uk Kim <jkim@FreeBSD.org>
uid Jung-uk Kim <jkim@niksun.com>
sub 4096g/B01CA5A0 2004-04-07
```

D.3.150. Zack Kirsch <zack@FreeBSD.org>

```
pub 1024D/1A725562 2010-11-05 Zack Kirsch <zack@freebsd.org>
    Key fingerprint = A8CC AA5E FB47 A386 E757 A2B8 BDD2 0684 1A72 5562
sub 1024g/6BFE2C06 2010-11-05
```

D.3.151. Jakub Klama <jceel@FreeBSD.org>

```
pub 2048R/2AAEA67D 2011-09-27
    Key fingerprint = 40D6 097A 174F 511B 80EB F3A3 0946 4193 2AAE A67D
uid Jakub Klama <jceel@FreeBSD.org>
sub 2048R/5291BC4D 2011-09-27
```

D.3.152. Andreas Klemm <andreas@FreeBSD.org>

```
pub 1024D/6C6F6CBA 2001-01-06 Andreas Klemm <andreas.klemm@eu.didata.com>
    Key fingerprint = F028 D51A 0D42 DD67 4109 19A3 777A 3E94 6C6F 6CBA
uid Andreas Klemm <andreas@klemm.gtn.com>
uid Andreas Klemm <andreas@FreeBSD.org>
uid Andreas Klemm <andreas@apsfilter.org>
sub 2048g/FE23F866 2001-01-06
```

D.3.153. Johann Kois <jkois@FreeBSD.org>

```
pub 1024D/DD61C2D8 2004-06-27 Johann Kois <J.Kois@web.de>
    Key fingerprint = 8B70 03DB 3C45 E71D 0ED4 4825 FEB0 EBEF DD61 C2D8
uid Johann Kois <jkois@freebsd.org>
sub 1024g/568307CB 2004-06-27
```

D.3.154. Sergei Kolobov <sergei@FreeBSD.org>

```
pub 1024D/3BA53401 2003-10-10 Sergei Kolobov <sergei@FreeBSD.org>
    Key fingerprint = A2F4 5F34 0586 CC9C 493A 347C 14EC 6E69 3BA5 3401
uid Sergei Kolobov <sergei@kolobov.com>
sub 2048g/F8243671 2003-10-10
```

D.3.155. Maxim Konovalov <maxim@FreeBSD.org>

```
pub 1024D/2C172083 2002-05-21 Maxim Konovalov <maxim@FreeBSD.org>
    Key fingerprint = 6550 6C02 EFC2 50F1 B7A3 D694 ECF0 E90B 2C17 2083
uid                                     Maxim Konovalov <maxim@macomnet.ru>
sub 1024g/F305DDCA 2002-05-21
```

D.3.156. Taras Korenko <taras@FreeBSD.org>

```
pub 1024D/8ACCC68B 2010-03-30
    Key fingerprint = 5128 2A8B 9BC1 A664 21E0 1E61 D838 54D3 8ACC C68B
uid                                     Taras Korenko <taras@freebsd.org>
uid                                     Taras Korenko <ds@ukrhub.net>
uid                                     Taras Korenko <tarasishche@gmail.com>
sub 2048g/8D7CC0FA 2010-03-30 [expires: 2015-03-29]
```

D.3.157. Joseph Koshy <jkoshy@FreeBSD.org>

```
pub 1024D/D93798B6 2001-12-21 Joseph Koshy (FreeBSD) <jkoshy@freebsd.org>
    Key fingerprint = 0DE3 62F3 EF24 939F 62AA 2E3D ABB8 6ED3 D937 98B6
sub 1024g/43FD68E9 2001-12-21
```

D.3.158. Wojciech A. Koszek <wkoszek@FreeBSD.org>

```
pub 1024D/C9F25145 2006-02-15
    Key fingerprint = 6E56 C571 9D33 D23E 9A61 8E50 623C AD62 C9F2 5145
uid                                     Wojciech A. Koszek <dunstan@FreeBSD.czyst.pl>
uid                                     Wojciech A. Koszek <wkoszek@FreeBSD.org>
sub 4096g/3BBD20A5 2006-02-15
```

D.3.159. Alex Kozlov <ak@FreeBSD.org>

```
pub 2048R/0D1D29A0 2012-03-01 [expires: 2024-02-27]
    Key fingerprint = 7774 4FCF 6AC9 126B BD0E DBF3 5EBF 4968 0D1D 29A0
uid                                     Alex Kozlov <ak@freebsd.org>
sub 2048R/2DD82C65 2012-03-01 [expires: 2024-02-27]
```

D.3.160. Steven Kreuzer <skreuzer@FreeBSD.org>

```
pub 1024D/E0D6F907 2009-03-16 [expires: 2013-04-25]
    Key fingerprint = 8D8F 14D6 ED9F 6BD0 7756 7A46 66BA B4B6 E0D6 F907
uid                                     Steven Kreuzer <skreuzer@exit2shell.com>
uid                                     Steven Kreuzer <skreuzer@freebsd.org>
```

D.3.161. Gábor Kövesdán <gabor@FreeBSD.org>

```
pub 1024D/2373A6B1 2006-12-05
    Key fingerprint = A42A 10D6 834B BEC0 26F0 29B1 902D D04F 2373 A6B1
uid      Gabor Kovesdan <gabor@FreeBSD.org>
sub 2048g/92B0A104 2006-12-05
```

D.3.162. Ana Kukec <anchie@FreeBSD.org>

```
pub 2048R/510D23BB 2010-04-18
    Key fingerprint = 0A9B 0ABB 0E1C B5A4 3408 398F 778A C3B4 510D 23BB
uid      Ana Kukec <anchie@FreeBSD.org>
sub 2048R/699E4DDA 2010-04-18
```

D.3.163. Roman Kurakin <rik@FreeBSD.org>

```
pub 1024D/C8550F4C 2005-12-16 [expires: 2008-12-15]
    Key fingerprint = 25BB 789A 6E07 E654 8E59 0FA9 42B1 937C C855 0F4C
uid      Roman Kurakin <rik@FreeBSD.org>
sub 2048g/D15F2AB6 2005-12-16 [expires: 2008-12-15]
```

D.3.164. Hideyuki KURASHINA <rushani@FreeBSD.org>

```
pub 1024D/439ADC57 2002-03-22 Hideyuki KURASHINA <rushani@bl.mmtr.or.jp>
    Key fingerprint = A052 6F98 6146 6FE3 91E2 DA6B F2FA 2088 439A DC57
uid      Hideyuki KURASHINA <rushani@FreeBSD.org>
uid      Hideyuki KURASHINA <rushani@jp.FreeBSD.org>
sub 1024g/64764D16 2002-03-22
```

D.3.165. Jun Kuriyama <kuriyama@FreeBSD.org>

```
pub 1024D/FE3B59CD 1998-11-23 Jun Kuriyama <kuriyama@imgsrc.co.jp>
    Key fingerprint = 5219 55CE AC84 C296 3A3B B076 EE3C 4DBB FE3B 59CD
uid      Jun Kuriyama <kuriyama@FreeBSD.org>
uid      Jun Kuriyama <kuriyama@jp.FreeBSD.org>
sub 2048g/1CF20D27 1998-11-23
```

D.3.166. René Ladan <rene@FreeBSD.org>

```
pub 1024D/E5642BFC 2008-11-03
    Key fingerprint = ADBC ECCD EB5F A6B4 549F 600D 8C9E 647A E564 2BFC
uid      René Ladan <rene@freebsd.org>
sub 2048g/C54EA560 2008-11-03
```

D.3.167. Julien Laffaye <jlaffaye@FreeBSD.org>

```
pub 2048R/6AEBE420 2011-06-06
    Key fingerprint = 031A B449 B383 5C3B B618 E2F4 BAD0 0F0E 6AEB E420
uid          Julien Laffaye <jlaffaye@FreeBSD.org>
sub 2048R/538B8D5B 2011-06-06
```

D.3.168. Clement Laforet <clement@FreeBSD.org>

```
pub 1024D/0723BA1D 2003-12-13 Clement Laforet (FreeBSD committer address) <clement@FreeBSD.org>
    Key fingerprint = 3638 4B14 8463 A67B DC7E 641C B118 5F8F 0723 BA1D
uid          Clement Laforet <sheepkiller@cultdeadsheep.org>
uid          Clement Laforet <clement.laforet@cotds.org>
sub 2048g/23D57658 2003-12-13
```

D.3.169. Max Laier <mlaier@FreeBSD.org>

```
pub 1024D/3EB6046D 2004-02-09
    Key fingerprint = 917E 7F25 E90F 77A4 F746 2E8D 5F2C 84A1 3EB6 046D
uid          Max Laier <max@love2party.net>
uid          Max Laier <max.laier@ira.uka.de>
uid          Max Laier <mlaier@freebsd.org>
uid          Max Laier <max.laier@tm.uka.de>
sub 4096g/EDD08B9B 2005-06-28
```

D.3.170. Erwin Lansing <erwin@FreeBSD.org>

```
pub 1024D/15256990 1998-07-03
    Key fingerprint = FB58 9797 299A F18E 2D3E 73D6 AB2F 5A5B 1525 6990
uid          Erwin Lansing <erwin@lansing.dk>
uid          Erwin Lansing <erwin@FreeBSD.org>
uid          Erwin Lansing <erwin@droso.dk>
uid          Erwin Lansing <erwin@droso.org>
uid          Erwin Lansing <erwin@aauug.dk>
sub 2048g/7C64013D 1998-07-03
```

D.3.171. Ganael Laplanche <martymac@FreeBSD.org>

```
pub 1024D/10B87391 2006-01-13
    Key fingerprint = D59D 984D 8988 7BB9 DA37 BA77 757E D5F0 10B8 7391
uid          Ganael LAPLANCHE <ganael.laplanche@martymac.org>
uid          Ganael LAPLANCHE <martymac@martymac.com>
uid          Ganael LAPLANCHE <ganael.laplanche@martymac.com>
uid          Ganael LAPLANCHE <martymac@martymac.org>
uid          Ganael LAPLANCHE <martymac@pasteur.fr>
uid          Ganael LAPLANCHE <ganael.laplanche@pasteur.fr>
```

```
uid          Ganael LAPLANCHE <martymac@FreeBSD.org>
sub 2048g/D65069D5 2006-01-13
```

D.3.172. Greg Larkin <glarkin@FreeBSD.org>

```
pub 1024D/1C940290 2003-10-09
   Key fingerprint = 8A4A 80AA F26C 8C2C D01B 94C6 D2C4 68B8 1C94 0290
uid          Greg Larkin (The FreeBSD Project) <glarkin@FreeBSD.org>
uid          Gregory C. Larkin (SourceHosting.Net, LLC) <glarkin@sourcehosting.net>
uid          [jpeg image of size 6695]
sub 2048g/47674316 2003-10-09
```

D.3.173. Frank J. Laszlo <laszlof@FreeBSD.org>

```
pub 4096R/012360EC 2006-11-06 [expires: 2011-11-05]
   Key fingerprint = 3D93 21DB B5CC 1339 E4B4 1BC4 AD50 C17C 0123 60EC
uid          Frank J. Laszlo <laszlof@FreeBSD.org>
```

D.3.174. Sam Lawrance <lawrance@FreeBSD.org>

```
pub 1024D/32708C59 2003-08-14
   Key fingerprint = 1056 2A02 5247 64D4 538D 6975 8851 7134 3270 8C59
uid          Sam Lawrance <lawrance@FreeBSD.org>
uid          Sam Lawrance <boris@brooknet.com.au>
sub 2048g/0F9CCF92 2003-08-14
```

D.3.175. Nate Lawson <njl@FreeBSD.org>

```
pub 1024D/60E5AC11 2007-02-07
   Key fingerprint = 18E2 7E5A FD6A 199B B08B E9FB 73C8 DB67 60E5 AC11
uid          Nate Lawson <nate@root.org>
sub 2048g/CDBC7E1B 2007-02-07
```

D.3.176. Jeremie Le Hen <jlh@FreeBSD.org>

```
pub 2048D/8BF6CF92 2012-04-18
   Key fingerprint = 66C9 B361 16CA BFF6 5C07 DA0A 28DE 3702 8BF6 CF92
uid          Jeremie Le Hen <jeremie@le-hen.org>
uid          Jeremie Le Hen <jeremie@lehen.org>
uid          Jeremie Le Hen <ttz@chchile.org>
uid          Jeremie Le Hen <jlh@FreeBSD.org>
sub 2048g/045479A3 2012-04-18
```

D.3.177. Yen-Ming Lee <leeym@FreeBSD.org>

```
pub 1024D/93FA8BD6 2007-05-21
    Key fingerprint = DEC4 6E7F 69C0 4AC3 21ED EE65 6C0E 9257 93FA 8BD6
uid          Yen-Ming Lee <leeym@leeym.com>
sub 2048g/899A3931 2007-05-21
```

D.3.178. Sam Leffler <sam@FreeBSD.org>

```
pub 1024D/BD147743 2005-03-28
    Key fingerprint = F618 F2FC 176B D201 D91C 67C6 2E33 A957 BD14 7743
uid          Samuel J. Leffler <sam@freebsd.org>
sub 2048g/8BA91D05 2005-03-28
```

D.3.179. Jean-Yves Lefort <jylefort@FreeBSD.org>

```
pub 1024D/A3B8006A 2002-09-07
    Key fingerprint = CC99 D1B0 8E44 293D 32F7 D92E CB30 FB51 A3B8 006A
uid          Jean-Yves Lefort <jylefort@FreeBSD.org>
uid          Jean-Yves Lefort <jylefort@brutele.be>
sub 4096g/C9271AFC 2002-09-07
```

D.3.180. Alexander Leidinger <netchild@FreeBSD.org>

```
pub 1024D/72077137 2002-01-31
    Key fingerprint = AA3A 8F69 B214 6BBD 5E73 C9A0 C604 3C56 7207 7137
uid          Alexander Leidinger <netchild@FreeBSD.org>
uid          [jpeg image of size 19667]
sub 2048g/8C9828D3 2002-01-31
```

D.3.181. Andrey V. Elsukov <ae@FreeBSD.org>

```
pub 2048R/10C8A17A 2010-05-29
    Key fingerprint = E659 1E1B 41DA 1516 F0C9 BC00 01C5 EA04 10C8 A17A
uid          Andrey V. Elsukov <ae@freebsd.org>
uid          Andrey V. Elsukov <bu7cher@yandex.ru>
sub 2048R/0F6D64C5 2010-05-29
```

D.3.182. Dejan Lesjak <lesi@FreeBSD.org>

```
pub 1024D/96C5221F 2004-08-18 Dejan Lesjak <lesi@FreeBSD.org>
    Key fingerprint = 2C5C 02EA 1060 1D6D 9982 38C0 1DA7 DBC4 96C5 221F
uid          Dejan Lesjak <dejan.lesjak@ijs.si>
sub 1024g/E0A69278 2004-08-18
```

D.3.183. Chuck Lever <cel@FreeBSD.org>

```
pub 1024D/8FFC2B87 2006-02-13
    Key fingerprint = 6872 923F 5012 F88B 394C 2F69 37B4 8171 8FFC 2B87
uid          Charles E. Lever <cel@freebsd.org>
sub 2048g/9BCE0459 2006-02-13
```

D.3.184. Greg Lewis <glewis@FreeBSD.org>

```
pub 1024D/1BB6D9E0 2002-03-05 Greg Lewis (FreeBSD) <glewis@FreeBSD.org>
    Key fingerprint = 2410 DA6D 5A3C D801 65FE C8DB DEEA 9923 1BB6 D9E0
uid          Greg Lewis <glewis@eyesbeyond.com>
sub 2048g/45E67D60 2002-03-05
```

D.3.185. Xin Li <delphij@FreeBSD.org>

```
pub 1024D/CAEEB8C0 2004-01-28
    Key fingerprint = 43B8 B703 B8DD 0231 B333 DC28 39FB 93A0 CAEE B8C0
uid          Xin LI <delphij@FreeBSD.org>
uid          Xin LI <delphij@frontfree.net>
uid          Xin LI <delphij@delphij.net>
uid          Xin LI <delphij@geekcn.org>

pub 1024D/42EA8A4B 2006-01-27 [expired: 2008-01-01]
    Key fingerprint = F19C 2616 FA97 9C13 2581 C6F3 85C5 1CCE 42EA 8A4B
uid          Xin LI <delphij@geekcn.org>
uid          Xin LI <delphij@FreeBSD.org>
uid          Xin LI <delphij@delphij.net>

pub 1024D/18EDEBA0 2008-01-02 [expired: 2010-01-02]
    Key fingerprint = 79A6 CF42 F917 DDCA F1C2 C926 8BEB DB04 18ED EBA0
uid          Xin LI <delphij@geekcn.org>
uid          Xin LI <delphij@FreeBSD.org>
uid          Xin LI <delphij@delphij.net>

pub 2048R/3FCA37C1 2010-01-10 [expired: 2012-01-10]
    Key fingerprint = 27EA 5D6C 9398 BA7F B205 8F70 04CE F812 3FCA 37C1
uid          Xin LI <delphij@delphij.net>
uid          Xin LI <delphij@gmail.com>
uid          Xin LI <delphij@geekcn.org>
uid          Xin LI <delphij@FreeBSD.org>

pub 4096R/2E54AB2C 2011-12-05
    Key fingerprint = D95C D3C3 8FA8 25C2 C62B 9FEA 0887 6D93 2E54 AB2C
uid          Xin Li <delphij@geekcn.org>
uid          Xin Li <delphij@delphij.net>
uid          Xin Li <delphij@FreeBSD.org>
sub 4096R/7832B740 2011-12-05
sub 2048R/BC50FBB3 2011-12-05 [expires: 2013-12-05]
sub 2048R/C894647D 2011-12-05 [expires: 2013-12-05]
```

D.3.186. Tai-hwa Liang <avatar@FreeBSD.org>

```
pub 1024R/F4013AB1 1998-05-13 Tai-hwa Liang <avatar@FreeBSD.org>
   Key fingerprint = 5B 05 1D 37 7F 35 31 4E 5D 38 BD 07 10 32 B9 D0
uid                               Tai-hwa Liang <avatar@mmlab.cse.yzu.edu.tw>
```

D.3.187. Ying-Chieh Liao <ijliao@FreeBSD.org>

```
pub 1024D/11C02382 2001-01-09 Ying-Chieh Liao <ijliao@CCCA.NCTU.edu.tw>
   Key fingerprint = 4E98 55CC 2866 7A90 EFD7 9DA5 ACC6 0165 11C0 2382
uid                               Ying-Chieh Liao <ijliao@FreeBSD.org>
uid                               Ying-Chieh Liao <ijliao@csie.nctu.edu.tw>
uid                               Ying-Chieh Liao <ijliao@dragon2.net>
uid                               Ying-Chieh Liao <ijliao@tw.FreeBSD.org>
sub 4096g/C1E16E89 2001-01-09
```

D.3.188. Ulf Lilleengen <lulf@FreeBSD.org>

```
pub 1024D/ADE1B837 2009-08-19 [expires: 2014-08-18]
   Key fingerprint = 3822 B4E6 6D1C 6F71 4AA8 7A27 ADDF C400 ADE1 B837
uid                               Ulf Lilleengen <lulf.lilleengen@gmail.com>
uid                               Ulf Lilleengen <lulf@pvv.ntnu.no>
uid                               Ulf Lilleengen <lulf@stud.ntnu.no>
uid                               Ulf Lilleengen <lulf@FreeBSD.org>
uid                               Ulf Lilleengen <lulf@idi.ntnu.no>
sub 2048g/B5409122 2009-08-19 [expires: 2014-08-18]
```

D.3.189. Clive Lin <clive@FreeBSD.org>

```
pub 1024D/A008C03E 2001-07-30 Clive Lin <clive@tongi.org>
   Key fingerprint = FA3F 20B6 A77A 6CEC 1856 09B0 7455 2805 A008 C03E
uid                               Clive Lin <clive@CirX.ORG>
uid                               Clive Lin <clive@FreeBSD.org>
sub 1024g/03C2DC87 2001-07-30 [expires: 2005-08-25]
```

D.3.190. Yi-Jheng Lin <yzlin@FreeBSD.org>

```
pub 2048R/A34C6A8A 2009-07-20
   Key fingerprint = 7E3A E981 BB7C 5D73 9534 ED39 0222 04D3 A34C 6A8A
uid                               Yi-Jheng Lin (FreeBSD) <yzlin@FreeBSD.org>
sub 2048R/B4D776FE 2009-07-20
```

D.3.191. Mark Linimon <linimon@FreeBSD.org>

```
pub 1024D/84C83473 2003-10-09
    Key fingerprint = 8D43 1B55 D127 0BFC 842E 1C96 803C 5A34 84C8 3473
uid      Mark Linimon <linimon@FreeBSD.org>
uid      Mark Linimon <linimon@lonesome.com>
sub 1024g/24BFF840 2003-10-09
```

D.3.192. Tilman Keskinözü <arved@FreeBSD.org>

```
pub 1024D/807AC53A 2002-06-03 [expires: 2013-09-07]
    Key fingerprint = A92F 344F 31A8 B8DE DDFA 7FB4 7C22 C39F 807A C53A
uid      Tilman Keskinözü <arved@arved.at>
uid      Tilman Keskinözü <arved@FreeBSD.org>
sub 1024g/FA351986 2002-06-03 [expires: 2013-09-07]
```

D.3.193. Dryice Liu <dryice@FreeBSD.org>

```
pub 1024D/77B67874 2005-01-28
    Key fingerprint = 8D7C F82D D28D 07E5 EF7F CD25 6B5B 78A8 77B6 7874
uid      Dryice Dong Liu (Dryice) <dryice@FreeBSD.org>
uid      Dryice Dong Liu (Dryice) <dryice@liu.com.cn>
uid      Dryice Dong Liu (Dryice) <dryice@hotpop.com>
uid      Dryice Dong Liu (Dryice) <dryiceliu@gmail.com>
uid      Dryice Dong Liu (Dryice) <dryice@dryice.name>
sub 2048g/ECFA49E4 2005-01-28
```

D.3.194. Tong Liu <nemoliu@FreeBSD.org>

```
pub 1024D/ECC7C907 2007-07-10
    Key fingerprint = B62E 3109 896B B283 E2FA 60FE A1BA F92E ECC7 C907
uid      Tong LIU <nemoliu@FreeBSD.org>
sub 4096g/B6D7B15D 2007-07-10
```

D.3.195. Zachary Loafman <zml@FreeBSD.org>

```
pub 1024D/4D65492D 2009-05-26
    Key fingerprint = E513 4AE9 5D6D 8BF9 1CD3 4389 4860 D79B 4D65 492D
uid      Zachary Loafman <zml@FreeBSD.org>
sub 2048g/1AD659F0 2009-05-26
```

D.3.196. Juergen Lock <nox@FreeBSD.org>

```
pub 1024D/1B6BF8FD 2006-12-22
    Key fingerprint = 33A7 7FAE 51AF 00BC F0D3 ECCE FAFD 34C1 1B6B FBFD
uid Juergen Lock <nox@FreeBSD.org>
sub 2048g/251229D1 2006-12-22
```

D.3.197. Remko Lodder <remko@FreeBSD.org>

```
pub 2048R/6EB8C8C8 2010-05-28 [expires: 2012-05-27]
    Key fingerprint = D692 91F9 F4EF D363 7F3F 4D17 9C75 DF7B 6EB8 C8C8
uid Remko Lodder (Remko Lodder's Key) <remko@FreeBSD.org>
sub 2048R/011C6AA0 2010-05-28 [expires: 2012-05-27]
```

D.3.198. Alexander Logvinov <avl@FreeBSD.org>

```
pub 1024D/1C47D5C0 2009-05-28
    Key fingerprint = 8B5F 880A 382B 075E E707 9DB2 E135 4176 1C47 D5C0
uid Alexander Logvinov <alexander@logvinov.com>
uid Alexander Logvinov (FreeBSD Ports Committer) <avl@FreeBSD.org>
uid Alexander Logvinov <ports@logvinov.com>
uid Alexander Logvinov <logvinov@gmail.com>
uid Alexander Logvinov <logvinov@yandex.ru>
sub 2048g/60BDD4BB 2009-05-28
```

D.3.199. Isabell Long <issy10@FreeBSD.org>

```
pub 4096R/EB83C2BD 2009-09-26
    Key fingerprint = D55A 42E7 0974 EFD9 3939 56B9 6E6B E425 EB83 C2BD
uid Isabell Long <isabell@issy10.co.uk>
uid Isabell Long <me@issy10.co.uk>
uid Isabell Long <isabell1121@gmail.com>
uid Isabell Long (BitFolk Ltd.) <isabell@bitfolk.com>
uid Isabell Long (College) <IL18685@woking.ac.uk>
uid Isabell Long (The Open University) <il948@my.open.ac.uk>
uid Isabell Long (Mailing lists address.) <lists@issy10.co.uk>
uid Isabell Long (YRS) <isabell@youngwiredstate.org>
uid Isabell Long (FreeBSD) <issy10@FreeBSD.org>
```

D.3.200. Scott Long <scottl@FreeBSD.org>

```
pub 1024D/017C5EBF 2003-01-18 Scott A. Long (This is my official FreeBSD key) <scottl@freebsd.org>
    Key fingerprint = 34EA BD06 44F7 F8C3 22BC B52C 1D3A F6D1 017C 5EBF
sub 1024g/F61C8F91 2003-01-18
```

D.3.201. Rick Macklem <rmacklem@FreeBSD.org>

```
pub 1024D/7FB9C5F1 2009-04-05
    Key fingerprint = B9EA 767A F6F3 3786 E0C7 434A 05C6 70D6 7FB9 C5F1
uid Rick Macklem <rmacklem@freebsd.org>
sub 1024g/D0B20E8A 2009-04-05
```

D.3.202. Bruce A. Mah <bmah@FreeBSD.org>

```
pub 1024D/5BA052C3 1997-12-08
    Key fingerprint = F829 B805 207D 14C7 7197 7832 D8CA 3171 5BA0 52C3
uid Bruce A. Mah <bmah@acm.org>
uid Bruce A. Mah <bmah@ca.sandia.gov>
uid Bruce A. Mah <bmah@ieee.org>
uid Bruce A. Mah <bmah@cisco.com>
uid Bruce A. Mah <bmah@employees.org>
uid Bruce A. Mah <bmah@freebsd.org>
uid Bruce A. Mah <bmah@packetdesign.com>
uid Bruce A. Mah <bmah@kitchenlab.org>
sub 2048g/B4E60EA1 1997-12-08
```

D.3.203. Ruslan Mahmatkhanov <rm@FreeBSD.org>

```
pub 2048R/F60D756F 2011-11-10
    Key fingerprint = 9D18 8A88 304C B78B 8003 0379 4574 0BAF F60D 756F
uid Ruslan Mahmatkhanov <rm@FreeBSD.org>
sub 2048R/B658C269 2011-11-10
```

D.3.204. Mike Makonnen <mtm@FreeBSD.org>

```
pub 1024D/7CD41F55 2004-02-06 Michael Telahun Makonnen <mtm@FreeBSD.Org>
    Key fingerprint = AC7B 5672 2D11 F4D0 EBF8 5279 5359 2B82 7CD4 1F55
uid Michael Telahun Makonnen <mtm@tmsa-inc.com>
uid Mike Makonnen <mtm@identd.net>
uid Michael Telahun Makonnen <mtm@acs-et.com>
sub 2048g/E7DC936B 2004-02-06
```

D.3.205. David Malone <dwmalone@FreeBSD.org>

```
pub 512/40378991 1994/04/21 David Malone <dwmalone@maths.tcd.ie>
    Key fingerprint = 86 A7 F4 86 39 2C 47 2C C1 C2 35 78 8E 2F B8 F5
```

D.3.206. Dmitry Marakasov <amdmi3@FreeBSD.org>

```
pub 1024D/F9D2F77D 2008-06-15 [expires: 2010-06-15]
    Key fingerprint = 55B5 0596 FF1E 8D84 5F56 9510 D35A 80DD F9D2 F77D
uid          Dmitry Marakasov <amdmi3@amdmi3.ru>
uid          Dmitry Marakasov <amdmi3@FreeBSD.org>
sub 2048g/2042CDD8 2008-06-15
```

D.3.207. Koop Mast <kwm@FreeBSD.org>

```
pub 1024D/F95426DA 2004-09-10 Koop Mast <kwm@rainbow-runner.nl>
    Key fingerprint = C66F 1835 0548 3440 8576 0FFE 6879 B7CD F954 26DA
uid          Koop Mast <kwm@FreeBSD.org>
sub 1024g/A782EEDD 2004-09-10
```

D.3.208. Cherry G. Mathew <cherry@FreeBSD.org>

```
pub 2048R/2D066FE1 2007-05-22
    Key fingerprint = FBF1 89FF 81BB E1C7 6C1B 378D 3438 20E9 2D06 6FE1
uid          Cherry G. Mathew (FreeBSD email) <cherry@FreeBSD.org>
uid          "Cherry G. Mathew" (NetBSD email) <cherry@NetBSD.org>
sub 2048R/7B2C4166 2007-05-22
```

D.3.209. Makoto Matsushita <matusita@FreeBSD.org>

```
pub 1024D/20544576 1999-04-18
    Key fingerprint = 71B6 13BF B262 2DD8 2B7C 6CD0 EB2D 4147 2054 4576
uid          Makoto Matsushita <matusita@matatabi.or.jp>
uid          Makoto Matsushita <matusita@FreeBSD.org>
uid          Makoto Matsushita <matusita@jp.FreeBSD.ORG>
uid          Makoto Matsushita <matusita@ist.osaka-u.ac.jp>
sub 1024g/F1F3C94D 1999-04-18
```

D.3.210. Martin Matuska <mm@FreeBSD.org>

```
pub 1024D/4261B0D1 2007-02-05
    Key fingerprint = 17C4 3F32 B3DE 3ED7 E84E 5592 A76B 8B03 4261 B0D1
uid          Martin Matuska <martin@matuska.org>
uid          Martin Matuska <mm@FreeBSD.org>
uid          Martin Matuska <martin.matuska@wu-wien.ac.at>
sub 2048g/3AC9A5A6 2007-02-05
```

D.3.211. Sergey Matveychuk <sem@FreeBSD.org>

```
pub 1024D/B71F605D 1999-10-13
    Key fingerprint = 4704 F374 DB28 BEC6 51C8 1322 4DC9 4BD8 B71F 605D
uid          Sergey Matveychuk <sem@FreeBSD.org>
uid          Sergey Matveychuk <sem@ciam.ru>
uid          Sergey Matveychuk <sem@core.inec.ru>
sub 2048g/DEAF9D91 1999-10-13
```

D.3.212. Tom McLaughlin <tmclaugh@FreeBSD.org>

```
pub 1024D/E2F7B3D8 2005-05-24
    Key fingerprint = 7692 B222 8D23 CF94 1993 0138 E339 E225 E2F7 B3D8
uid          Tom McLaughlin (Personal email address) <tmclaugh@sdf.lonestar.org>
uid          Tom McLaughlin (Work email address) <tmclaughlin@meditech.com>
uid          Tom McLaughlin (FreeBSD email address) <tmclaugh@FreeBSD.org>
sub 2048g/16838F62 2005-05-24
```

D.3.213. Jean Milanez Melo <jmelo@FreeBSD.org>

```
pub 1024D/AA5114BF 2006-03-03
    Key fingerprint = 826D C2AA 6CF2 E29A EBE7 4776 D38A AB83 AA51 14BF
uid          Jean Milanez Melo <jmelo@FreeBSD.org>
uid          Jean Milanez Melo <jmelo@freebsdbrasil.com.br>
sub 4096g/E9E1CBD9 2006-03-03
```

D.3.214. Kenneth D. Merry <ken@FreeBSD.org>

```
pub 1024D/54C745B5 2000-05-15 Kenneth D. Merry <ken@FreeBSD.org>
    Key fingerprint = D25E EBC5 F17A 9E52 84B4 BF14 9248 F0DA 54C7 45B5
uid          Kenneth D. Merry <ken@kdm.org>
sub 2048g/89D0F797 2000-05-15

pub 1024R/2FA0A505 1995-10-30 Kenneth D. Merry <ken@plutotech.com>
    Key fingerprint = FD FA 85 85 95 C4 8E E8 98 1A CA 18 56 F0 00 1F
```

D.3.215. Dirk Meyer <dinoex@FreeBSD.org>

```
pub 1024R/331CDA5D 1995-06-04 Dirk Meyer <dinoex@FreeBSD.org>
    Key fingerprint = 44 16 EC 0A D3 3A 4F 28 8A 8A 47 93 F1 CF 2F 12
uid          Dirk Meyer <dirk.meyer@dinoex.sub.org>
uid          Dirk Meyer <dirk.meyer@guug.de>
```

D.3.216. Yoshiro Sanpei MIHIRA <sanpei@FreeBSD.org>

```
pub 1024R/391C5D69 1996-11-21 sanpei@SEAPLE.ICC.NE.JP
    Key fingerprint = EC 04 30 24 B0 6C 1E 63 5F 5D 25 59 3E 83 64 51
uid      MIHIRA Yoshiro <sanpei@sanpei.org>
uid      Yoshiro MIHIRA <sanpei@FreeBSD.org>
uid      MIHIRA Yoshiro <sanpei@yy.cs.keio.ac.jp>
uid      MIHIRA Yoshiro <sanpei@cc.keio.ac.jp>
uid      MIHIRA Yoshiro <sanpei@educ.cc.keio.ac.jp>
uid      MIHIRA Yoshiro <sanpei@st.keio.ac.jp>
```

D.3.217. Robert Millan <rmh@FreeBSD.org>

```
pub 4096R/DEA2C38E 2009-08-14
    Key fingerprint = A537 F029 AAAE 0E9C 39A7 C22C BB9D 98D9 DEA2 C38E
uid      Robert Millan <rmh@debian.org>
uid      Robert Millan <rmh@freebsd.org>
uid      Robert Millan <rmh@gnu.org>
sub 4096R/65A0A9CE 2009-08-14
sub 4096R/41F37946 2009-08-14
```

D.3.218. Stephen Montgomery-Smith <stephen@FreeBSD.org>

```
pub 2048R/9A92D807 2011-06-14
    Key fingerprint = 2B61 D82E 168E F08B 6E08 712E 2DF1 2BD1 9A92 D807
uid      Stephen Montgomery-Smith <stephen@freebsd.org>
sub 2048R/A4BA6560 2011-06-14
```

D.3.219. Marcel Moolenaar <marcel@FreeBSD.org>

```
pub 1024D/61EE89F6 2002-02-09 Marcel Moolenaar <marcel@xcllnt.net>
    Key fingerprint = 68BB E2B7 49AA FF69 CA3A DF71 A605 A52D 61EE 89F6
sub 1024g/6EAAB456 2002-02-09
```

D.3.220. Kris Moore <kmoore@FreeBSD.org>

```
pub 1024D/6294612C 2009-05-26
    Key fingerprint = 8B70 9876 346F 1F97 5687 6950 4C92 D789 6294 612C
uid      Kris Moore <kmoore@freebsd.org>
sub 2048g/A7FFE8FB 2009-05-26
```

D.3.221. Dmitry Morozovsky <marck@FreeBSD.org>

```
pub 1024D/6B691B03 2001-07-20
    Key fingerprint = 39AC E336 F03D C0F8 5305 B725 85D4 5045 6B69 1B03
uid          Dmitry Morozovsky <marck@rinet.ru>
uid          Dmitry Morozovsky <marck@FreeBSD.org>
sub 2048g/44D656F8 2001-07-20
```

D.3.222. Alexander Motin <mav@FreeBSD.org>

```
pub 1024D/0577BACA 2007-04-20 [expires: 2012-04-18]
    Key fingerprint = 0E84 B263 E97D 3E48 161B 98A2 D240 A09E 0577 BACA
uid          Alexander Motin <mav@freebsd.org>
uid          Alexander Motin <mav@mavhome.dp.ua>
uid          Alexander Motin <mav@alkar.net>
sub 2048g/4D59D1C2 2007-04-20 [expires: 2012-04-18]
```

D.3.223. Felipe de Meirelles Motta <lippe@FreeBSD.org>

```
pub 1024D/F2CF7DAE 2008-09-02 [expires: 2010-09-02]
    Key fingerprint = 0532 A900 286D DAFD 099D 394D 231B AF20 F2CF 7DAE
uid          Felipe de Meirelles Motta (FreeBSD Ports Committer) <lippe@FreeBSD.org>
sub 2048g/38E8EEF3 2008-09-02 [expires: 2010-09-02]
```

D.3.224. Rich Murphey <rich@FreeBSD.org>

```
pub 1024R/583443A9 1995-03-31 Rich Murphey <rich@lamprey.utmb.edu>
    Key fingerprint = AF A0 60 C4 84 D6 0C 73 D1 EF C0 E9 9D 21 DB E4
```

D.3.225. Akinori MUSHASHA <knu@FreeBSD.org>

```
pub 1024D/9FD9E1EE 2000-03-21 Akinori MUSHASHA <knu@and.or.jp>
    Key fingerprint = 081D 099C 1705 861D 4B70 B04A 920B EFC7 9FD9 E1EE
uid          Akinori MUSHASHA <knu@FreeBSD.org>
uid          Akinori MUSHASHA <knu@idaemons.org>
uid          Akinori MUSHASHA <knu@ruby-lang.org>
sub 1024g/71BA9D45 2000-03-21
```

D.3.226. Thomas Möstl <tmm@FreeBSD.org>

```
pub 1024D/419C776C 2000-11-28 Thomas Moestl <tmm@FreeBSD.org>
    Key fingerprint = 1C97 A604 2BD0 E492 51D0 9C0F 1FE6 4F1D 419C 776C
uid          Thomas Moestl <tmoestl@gmx.net>
uid          Thomas Moestl <t.moestl@tu-bs.de>
```

sub 2048g/ECE63CE6 2000-11-28

D.3.227. Masafumi NAKANE <max@FreeBSD.org>

```
pub 1024D/CE356B59 2000-02-19 Masafumi NAKANE <max@wide.ad.jp>
    Key fingerprint = EB40 BCAB 4CE5 0764 9942 378C 9596 159E CE35 6B59
uid                               Masafumi NAKANE <max@FreeBSD.org>
uid                               Masafumi NAKANE <max@accessibility.org>
uid                               Masafumi NAKANE <kd5pdi@qsl.net>
sub 1024g/FA9BD48B 2000-02-19
```

D.3.228. Maho Nakata <maho@FreeBSD.org>

```
pub 1024D/F28B4069 2009-02-09
    Key fingerprint = 3FE4 99A9 6F41 8161 4F5F 240C 8615 A60C F28B 4069
uid                               Maho NAKATA (NAKATA's FreeBSD.org alias) <maho@FreeBSD.org>
sub 2048g/6B49098E 2009-02-09
```

D.3.229. Yoichi NAKAYAMA <yoichi@FreeBSD.org>

```
pub 1024D/E0788E46 2000-12-28 Yoichi NAKAYAMA <yoichi@assist.media.nagoya-u.ac.jp>
    Key fingerprint = 1550 2662 46B3 096C 0460 BC03 800D 0C8A E078 8E46
uid                               Yoichi NAKAYAMA <yoichi@eken.phys.nagoya-u.ac.jp>
uid                               Yoichi NAKAYAMA <yoichi@FreeBSD.org>
sub 1024g/B987A394 2000-12-28
```

D.3.230. Edward Tomasz Napierala <trasz@FreeBSD.org>

```
pub 1024D/8E53F00E 2007-04-13
    Key fingerprint = DD8F 91B0 12D9 6237 42D9 DBE1 AFC8 CDE9 8E53 F00E
uid                               Edward Tomasz Napierala <trasz@FreeBSD.org>
sub 2048g/7C1F5D67 2007-04-13
```

D.3.231. Alexander Nedotsukov <bland@FreeBSD.org>

```
pub 1024D/D004116C 2003-08-14 Alexander Nedotsukov <bland@FreeBSD.org>
    Key fingerprint = 35E2 5020 55FC 2071 4ADD 1A4A 86B6 8A5D D004 116C
sub 1024g/1CCA8D46 2003-08-14
```

D.3.232. George V. Neville-Neil <gnn@FreeBSD.org>

```
pub 1024D/440A33D2 2002-09-17
    Key fingerprint = AF66 410F CC8D 1FC9 17DB 6225 61D8 76C1 440A 33D2
uid      George V. Neville-Neil <gnn@freebsd.org>
uid      George V. Neville-Neil <gnn@neville-neil.com>
sub 2048g/95A74F6E 2002-09-17
```

D.3.233. Simon L. Nielsen <simon@FreeBSD.org>

```
pub 1024D/FF7490AB 2007-01-14
    Key fingerprint = 4E92 BA8D E45E 85E2 0380 B264 049C 7480 FF74 90AB
uid      Simon L. Nielsen <simon@FreeBSD.org>
uid      Simon L. Nielsen <simon@nitro.dk>
sub 2048g/E3F5A76E 2007-01-14
```

D.3.234. Robert Noland <rnoland@FreeBSD.org>

```
pub 1024D/8A9F44E3 2007-07-24
    Key fingerprint = 107A 0C87 E9D0 E581 677B 2A28 3384 EB43 8A9F 44E3
uid      Robert C. Noland III <rnoland@FreeBSD.org>
uid      Robert C. Noland III (Personal Key) <rnoland@2hip.net>
sub 2048g/76C3CF00 2007-07-24
```

D.3.235. Anders Nordby <anders@FreeBSD.org>

```
pub 1024D/00835956 2000-08-13 Anders Nordby <anders@fix.no>
    Key fingerprint = 1E0F C53C D8DF 6A8F EAAD 19C5 D12A BC9F 0083 5956
uid      Anders Nordby <anders@FreeBSD.org>
sub 2048g/4B160901 2000-08-13
```

D.3.236. Michael Nottebrock <lofi@FreeBSD.org>

```
pub 1024D/6B2974B0 2002-06-06 Michael Nottebrock <michaelnottebrock@gmx.net>
    Key fingerprint = 1079 3C72 0726 F300 B8EC 60F9 5E17 3AF1 6B29 74B0
uid      Michael Nottebrock <lofi@freebsd.org>
uid      Michael Nottebrock <lofi@tigress.com>
uid      Michael Nottebrock <lofi@lofi.dyndns.org>
uid      Michael Nottebrock <michaelnottebrock@web.de>
uid      Michael Nottebrock <michaelnottebrock@meitner.wh.uni-dortmund.de>
sub 1024g/EF652E04 2002-06-06 [expires: 2004-06-15]
```

D.3.237. David O'Brien <obrien@FreeBSD.org>

```
pub 1024R/34F9F9D5 1995-04-23 David E. O'Brien <defunct - obrien@Sea.Legent.com>
    Key fingerprint = B7 4D 3E E9 11 39 5F A3 90 76 5D 69 58 D9 98 7A
uid                                David E. O'Brien <obrien@Nuxi.com>
uid                                deobrien@ucdavis.edu
uid                                David E. O'Brien <whois Do38>
uid                                David E. O'Brien <obrien@FreeBSD.org>
uid                                David E. O'Brien <dobrien@seas.gwu.edu>
uid                                David E. O'Brien <obrien@cs.ucdavis.edu>
uid                                David E. O'Brien <defunct - obrien@media.sra.com>
uid                                David E. O'Brien <obrien@elsewhere.roanoke.va.us>
uid                                David E. O'Brien <obrien@Nuxi.com>

pub 1024D/7F9A9BA2 1998-06-10 "David E. O'Brien" <obrien@cs.ucdavis.edu>
    Key fingerprint = 02FD 495F D03C 9AF2 5DB7 F496 6FC8 DABD 7F9A 9BA2
uid                                "David E. O'Brien" <obrien@Nuxi.com>
uid                                "David E. O'Brien" <obrien@FreeBSD.org>
sub 3072g/BA32C20D 1998-06-10
```

D.3.238. Philip Paeps <philip@FreeBSD.org>

```
pub 4096R/C5D34D05 2006-10-22
    Key fingerprint = 356B AE02 4763 F739 2FA2 E438 2649 E628 C5D3 4D05
uid                                Philip Paeps <philip@paeps.cx>
uid                                Philip Paeps <philip@nixsys.be>
uid                                Philip Paeps <philip@fosdem.org>
uid                                Philip Paeps <philip@freebsd.org>
uid                                Philip Paeps <philip@pub.telenet.be>
sub 1024D/035EFC58 2006-10-22 [expires: 2010-10-13]
sub 2048g/6E5FD7D6 2006-10-22 [expires: 2010-10-14]
```

D.3.239. Josh Paetzel <jpaetzel@FreeBSD.org>

```
pub 1024D/27AFAECB 2007-05-11
    Key fingerprint = 8A48 EF36 5E9F 4EDA 5A8C 11B4 26F9 01F1 27AF AECB
uid                                Josh Paetzel (BSD UNIX) <josh@tcbug.org>
uid                                Josh Paetzel <josh@rephunter.net>
uid                                Josh Paetzel <josh@pcbsd.org>
uid                                Josh Paetzel <jpaetzel@FreeBSD.org>
sub 2048g/E0F5996B 2007-05-11
```

D.3.240. Gábor Páli <pgj@FreeBSD.org>

```
pub 1024D/9E3F9BE6 2008-04-17 [expires: 2013-04-16]
    Key fingerprint = DA0B 2143 0FC8 EE5F E211 D329 7D4B 6E18 9E3F 9BE6
uid                                Gabor PALI <pgj@FreeBSD.org>
uid                                PÁLI Gábor János <pali.gabor@gmail.com>
```

sub 2048g/A780C60B 2008-04-17 [expires: 2013-04-16]

D.3.241. Hiten Pandya <hmp@FreeBSD.org>

```
pub 1024D/938CACA8 2004-02-13 Hiten Pandya (FreeBSD) <hmp@FreeBSD.org>
    Key fingerprint = 84EB C75E C75A 50ED 304E E446 D974 7842 938C ACA8
uid                                     Hiten Pandya <hmp@backplane.com>
sub 2048g/783874B5 2004-02-13
```

D.3.242. Dima Panov <fluffy@FreeBSD.org>

```
pub 1024D/93E3B018 2006-11-08
    Key fingerprint = C73E 2B72 1FFD 61BD E206 1234 A626 76ED 93E3 B018
uid                                     Dima Panov (FreeBSD.ORG Committer) <fluffy@FreeBSD.ORG>
uid                                     Dima Panov (at home) <Fluffy@Fluffy.Khv.RU>
uid                                     Dima Panov (at home) <fluffy.khv@gmail.com>
sub 2048g/89047419 2006-11-08

pub 4096R/D5398F29 2009-08-09
    Key fingerprint = 2D30 2CCB 9984 130C 6F87 BAFC FB8B A09D D539 8F29
uid                                     Dima Panov (FreeBSD.ORG Committer) <fluffy@FreeBSD.ORG>
uid                                     Dima Panov (at Home) <fluffy@Fluffy.Khv.RU>
uid                                     Dima Panov (at GMail) <fluffy.khv@gmail.com>
sub 4096R/915A7785 2009-08-09
```

D.3.243. Andrew Pantyukhin <sat@FreeBSD.org>

```
pub 1024D/6F38A569 2006-05-06
    Key fingerprint = 4E94 994A C2EF CB86 C144 3B04 3381 67C0 6F38 A569
uid                                     Andrew Pantyukhin <infofarmer@gubkin.ru>
uid                                     Andrew Pantyukhin <sat@FreeBSD.org>
uid                                     Andrew Pantyukhin <infofarmer@gmail.com>
uid                                     Andrew Pantyukhin <infofarmer@mail.ru>
sub 2048g/5BD4D469 2006-05-06
```

D.3.244. Navdeep Parhar <np@FreeBSD.org>

```
pub 1024D/ACAB8812 2009-06-08
    Key fingerprint = C897 7AFB AFC0 4DA9 7B76 D991 CAB2 2B93 ACAB 8812
uid                                     Navdeep Parhar <np@FreeBSD.org>
sub 2048g/AB61D2DC 2009-06-08
```

D.3.245. Rui Paulo <rpaulo@FreeBSD.org>

```
pub 4096R/39CB4153 2010-02-03
    Key fingerprint = ABE8 8465 DE8F F04D E9C8 3FF6 AF89 B2E6 39CB 4153
uid          Rui Paulo <rpaulo@FreeBSD.org>
uid          Rui Paulo <rpaulo@gmail.com>
sub 4096R/F87D2F34 2010-02-03
```

D.3.246. Mark Peek <mp@FreeBSD.org>

```
pub 1024D/330D4D01 2002-01-27 Mark Peek <mp@FreeBSD.org>
    Key fingerprint = 510C 96EE B4FB 1B0A 2CF8 A0AF 74B0 0B0E 330D 4D01
sub 1024g/9C6CAC09 2002-01-27
```

D.3.247. Peter Pentchev <roam@FreeBSD.org>

```
pub 1024D/16194553 2002-02-01
    Key fingerprint = FDBA FD79 C26F 3C51 C95E DF9E ED18 B68D 1619 4553
uid          Peter Pentchev <roam@ringlet.net>
uid          Peter Pentchev <roam@cnsys.bg>
uid          Peter Pentchev <roam@sbnd.net>
uid          Peter Pentchev <roam@online.bg>
uid          Peter Pentchev <roam@orbitel.bg>
uid          Peter Pentchev <roam@FreeBSD.org>
uid          Peter Pentchev <roam@techlab.officel.bg>
uid          Peter Pentchev <roam@hoster.bg>
uid          Peter Pentchev <roam@space.bg>
sub 1024g/7074473C 2002-02-01

pub 4096R/2527DF13 2009-10-16
    Key fingerprint = 2EE7 A7A5 17FC 124C F115 C354 651E EFB0 2527 DF13
uid          Peter Pentchev <roam@ringlet.net>
uid          Peter Pentchev <roamer@users.sourceforge.net>
uid          Peter Pentchev <roam@cpan.org>
uid          Peter Pentchev <roam@cnsys.bg>
uid          Peter Pentchev <roam@sbnd.net>
uid          Peter Pentchev <roam@online.bg>
uid          Peter Pentchev <roam@orbitel.bg>
uid          Peter Pentchev <roam@FreeBSD.org>
uid          Peter Pentchev <roam@techlab.officel.bg>
uid          Peter Pentchev <roam@hoster.bg>
uid          Peter Pentchev <roam@space.bg>
uid          Peter Pentchev <roam-guest@alioth.debian.org>
uid          Peter Pentchev <ppentchev@alumni.princeton.edu>
sub 4096R/D0B337AA 2009-10-16
```

D.3.248. Denis Peplin <den@FreeBSD.org>

```
pub 1024D/485DDDF5 2003-09-11 Denis Peplin <den@FreeBSD.org>
    Key fingerprint = 495D 158C 8EC9 C2C1 80F5 EA96 6F72 7C1C 485D DDF5
sub 1024g/E70BA158 2003-09-11
```

D.3.249. Christian S.J. Peron <csjp@FreeBSD.org>

```
pub 1024D/033FA33C 2009-05-16
    Key fingerprint = 74AA 6040 89A7 936E D970 DDC0 CC71 6954 033F A33C
uid Christian S.J. Peron <csjp@FreeBSD.ORG>
sub 2048g/856B194A 2009-05-16
```

D.3.250. Gerald Pfeifer <gerald@FreeBSD.org>

```
pub 1024D/745C015A 1999-11-09 Gerald Pfeifer <gerald@pfeifer.com>
    Key fingerprint = B215 C163 3BCA 0477 615F 1B35 A5B3 A004 745C 015A
uid Gerald Pfeifer <Gerald.Pfeifer@vibe.at>
uid Gerald Pfeifer <pfeifer@dbai.tuwien.ac.at>
uid Gerald Pfeifer <gerald@pfeifer.at>
uid Gerald Pfeifer <gerald@FreeBSD.org>
sub 1536g/F0156927 1999-11-09
```

D.3.251. Giuseppe Pilichi <jacula@FreeBSD.org>

```
pub 4096R/8B9F4B8B 2006-03-08
    Key fingerprint = 31AD 73AE 0EC0 16E5 4108 8391 D942 5F20 8B9F 4B8B
uid Giuseppe Pilichi (Jacula Modyun) <jacula@FreeBSD.org>
uid Giuseppe Pilichi (Jacula Modyun) <jaculamodyun@gmail.com>
uid Giuseppe Pilichi (Jacula Modyun) <gpilch@gmail.com>
uid Giuseppe Pilichi (Jacula Modyun) <jacula@gmail.com>
sub 4096R/FB4D05A3 2006-03-08
```

D.3.252. John Polstra <jdp@FreeBSD.org>

```
pub 1024R/BFBCF449 1997-02-14 John D. Polstra <jdp@polstra.com>
    Key fingerprint = 54 3A 90 59 6B A4 9D 61 BF 1D 03 09 35 8D F6 0D
```

D.3.253. Kirill Ponomarew <krion@FreeBSD.org>

```
pub 1024D/AEB426E5 2002-04-07
    Key fingerprint = 58E7 B953 57A2 D9DD 4960 2A2D 402D 46E9 AEB4 26E5
uid Kirill Ponomarew <krion@voodoo.bawue.com>
uid Kirill Ponomarew <krion@guug.de>
```

```
uid          Kirill Ponomarew <krion@FreeBSD.org>
sub 1024D/05AC7CA0 2006-01-30 [expires: 2008-01-30]
sub 2048g/C3EE5537 2006-01-30 [expires: 2008-01-30]
```

D.3.254. Stephane E. Potvin <sepotvin@FreeBSD.org>

```
pub 1024D/3097FE7B 2002-08-06
   Key fingerprint = 6B56 62FA ADE1 6F46 BB62 8B1C 99D3 97B5 3097 FE7B
uid          Stephane E. Potvin <sepotvin@videotron.ca>
uid          Stephane E. Potvin <stephane.potvin@telcobridges.com>
uid          Stephane E. Potvin <stephane_potvin@telcobridges.com>
uid          Stephane E. Potvin <sepotvin@FreeBSD.org>
sub 2048g/0C427BC9 2002-08-06
```

D.3.255. Mark Pulford <markp@FreeBSD.org>

```
pub 1024D/182C368F 2000-05-10 Mark Pulford <markp@FreeBSD.org>
   Key fingerprint = 58C9 C9BF C758 D8D4 7022 8EF5 559F 7F7B 182C 368F
uid          Mark Pulford <mark@kyne.com.au>
sub 2048g/380573E8 2000-05-10
```

D.3.256. Alejandro Pulver <alepulver@FreeBSD.org>

```
pub 1024D/945C3F61 2005-11-13
   Key fingerprint = 085F E8A2 4896 4B19 42A4 4179 895D 3912 945C 3F61
uid          Alejandro Pulver (Ale's GPG key pair) <alepulver@FreeBSD.org>
uid          Alejandro Pulver (Ale's GPG key pair) <alejandro@varnet.biz>
sub 2048g/6890C6CA 2005-11-13
```

D.3.257. Thomas Quinot <thomas@FreeBSD.org>

```
pub 1024D/393D2469 1999-09-23 Thomas Quinot <thomas@cuivre.fr.eu.org>
   Empreinte de la clé = 4737 A0AD E596 6D30 4356 29B8 004D 54B8 393D 2469
uid          Thomas Quinot <thomas@debian.org>
uid          Thomas Quinot <thomas@FreeBSD.org>
sub 1024g/8DE13BB2 1999-09-23
```

D.3.258. Herve Quiroz <hq@FreeBSD.org>

```
pub 1024D/85AC8A80 2004-07-22 Herve Quiroz <hq@FreeBSD.org>
   Key fingerprint = 14F5 BC56 D736 102D 41AF A07B 1D97 CE6C 85AC 8A80
uid          Herve Quiroz <herve.quiroz@esil.univ-mrs.fr>
sub 1024g/8ECCAFED 2004-07-22
```

D.3.259. Doug Rabson <dfr@FreeBSD.org>

```
pub 1024D/59F57821 2004-02-07
    Key fingerprint = 9451 C4FE 1A7E 117B B95F 1F8F B123 456E 59F5 7821
uid          Doug Rabson <dfr@nlsystems.com>
sub 1024g/6207AA32 2004-02-07
```

D.3.260. Lars Balkar Rasmussen <lbr@FreeBSD.org>

```
pub 1024D/9EF6F27F 2006-04-30
    Key fingerprint = F251 28B7 897C 293E 04F8 71EE 4697 F477 9EF6 F27F
uid          Lars Balkar Rasmussen <lbr@FreeBSD.org>
sub 2048g/A8C1CFD4 2006-04-30
```

D.3.261. Chris Rees <crees@FreeBSD.org>

```
pub 2048R/E1EBCAC4 2011-06-11 [expires: 2012-06-10]
    Key fingerprint = 2066 B855 E4B1 226A 8B4C B6E6 B084 92D1 E1EB CAC4
uid          Chris Rees <crees@freebsd.org>
sub 2048R/FA45D5D6 2011-06-11 [expires: 2012-06-10]
```

D.3.262. Jim Rees <rees@FreeBSD.org>

```
pub 512/B623C791 1995/02/21 Jim Rees <rees@umich.edu>
    Key fingerprint = 02 5F 1B 15 B4 6E F1 3E F1 C5 E0 1D EA CC 17 88
```

D.3.263. Benedict Reuschling <bcr@FreeBSD.org>

```
pub 1024D/4A819348 2009-05-24
    Key fingerprint = 2D8C BDF9 30FA 75A5 A0DF D724 4D26 502E 4A81 9348
uid          Benedict Reuschling <bcr@FreeBSD.org>
sub 2048g/8DA16EDD 2009-05-24
```

D.3.264. Tom Rhodes <trhodes@FreeBSD.org>

```
pub 1024D/FB7D88E1 2008-05-07
    Key fingerprint = 8279 3100 2DF2 F00E 7FDD AC2C 5776 23AB FB7D 88E1
uid          Tom Rhodes (trhodes) <trhodes@FreeBSD.org>
sub 4096g/7B0CD79F 2008-05-07
```

D.3.265. Benno Rice <benno@FreeBSD.org>

```
pub 1024D/87C59909 2002-01-16 Benno Rice <benno@FreeBSD.org>
    Key fingerprint = CE27 DADA 08E3 FAA3 88F1 5B31 5E34 705A 87C5 9909
uid                               Benno Rice <benno@jeamland.net>
sub 1024g/4F7C2BAD 2002-01-16 [expires: 2007-01-15]
```

D.3.266. Beech Rintoul <beech@FreeBSD.org>

```
pub 1024D/ECBFDC44 2011-08-29
    Key fingerprint = 6921 47CC 8B61 7C02 70EF 4D00 16B4 EAB7 ECBF DC44
uid                               Beech Rintoul <beech@freebsd.org>
sub 1024g/F1FD1C3D 2011-08-29
```

D.3.267. Matteo Rionato <matteo@FreeBSD.org>

```
pub 1024D/1EC56BEC 2003-01-05 [expires: 2009-09-07]
    Key fingerprint = F0F3 1B43 035D 65B1 08E9 4D66 D8CA 78A5 1EC5 6BEC
uid                               Matteo Rionato (Rionda) <matteo@FreeBSD.ORG>
uid                               Matteo Rionato (Rionda) <rionda@riondabsd.net>
uid                               Matteo Rionato (Rionda) <rionda@gufi.org>
uid                               Matteo Rionato (Rionda) <matteo@rionato.com>
uid                               Matteo Rionato (Rionda) <rionda@rionato.com>
uid                               Matteo Rionato (Rionda) <rionda@FreeSBIE.ORG>
uid                               Matteo Rionato (Rionda) <rionda@autistici.org>
sub 2048g/87C44A55 2008-09-23 [expires: 2009-09-23]
```

D.3.268. Ollivier Robert <roberto@FreeBSD.org>

```
pub 1024D/7DCAE9D3 1997-08-21
    Key fingerprint = 2945 61E7 D4E5 1D32 C100 DBEC A04F FB1B 7DCA E9D3
uid                               Ollivier Robert <roberto@keltia.freenix.fr>
uid                               Ollivier Robert <roberto@FreeBSD.org>
sub 2048g/C267084D 1997-08-21
```

D.3.269. Craig Rodrigues <rodrigc@FreeBSD.org>

```
pub 1024D/3998479D 2005-05-20
    Key fingerprint = F01F EBE6 F5C8 6DC2 954F 098F D20A 8A2A 3998 479D
uid                               Craig Rodrigues <rodrigc@freebsd.org>
uid                               Craig Rodrigues <rodrigc@crodrigues.org>
sub 2048g/AA77E09B 2005-05-20
```

D.3.270. Guido van Rooij <guido@FreeBSD.org>

```
pub 1024R/599F323D 1996-05-18 Guido van Rooij <guido@gvr.org>
   Key fingerprint = 16 79 09 F3 C0 E4 28 A7 32 62 FA F6 60 31 C0 ED
uid                               Guido van Rooij <guido@gvr.win.tue.nl>

pub 1024D/A95102C1 2000-10-25 Guido van Rooij <guido@madison-gurkha.nl>
   Key fingerprint = 5B3E 51B7 0E7A D170 0574 1E51 2471 117F A951 02C1
uid                               Guido van Rooij <guido@madison-gurkha.com>
sub 1024g/A5F20553 2000-10-25
```

D.3.271. Eygene Ryabinkin <rea@FreeBSD.org>

```
pub 3072D/8152ECFB 2010-10-27
   Key fingerprint = 82FE 06BC D497 C0DE 49EC 4FF0 16AF 9EAE 8152 ECFB
uid                               Eygene Ryabinkin <rea-fbsd@codelabs.ru>
uid                               Eygene Ryabinkin <rea@freebsd.org>
uid                               Eygene Ryabinkin <rea@codelabs.ru>
sub 3072g/5FC03749 2010-10-27
```

D.3.272. Aleksandr Rybalko <ray@FreeBSD.org>

```
pub 2048R/4B7B7A4E 2011-05-24
   Key fingerprint = BB9F D01D 7327 0B33 B2F5 6C72 EC49 E6ED 4B7B 7A4E
uid                               Aleksandr Rybalko (Aleksandr Rybalko FreeBSD project identification) <ray@fr
sub 2048R/99F9F9EF 2011-05-24
```

D.3.273. Niklas Saers <niklas@FreeBSD.org>

```
pub 1024D/C822A476 2004-03-09 Niklas Saers <niklas@saers.com>
   Key fingerprint = C41E F734 AF0E 3D21 7499 9EB1 9A31 2E7E C822 A476
sub 1024g/81E2FF36 2004-03-09
```

D.3.274. Boris Samorodov <bsam@FreeBSD.org>

```
pub 1024D/ADFD5C9A 2006-06-21
   Key fingerprint = 81AA FED0 6050 208C 0303 4007 6C03 7263 ADFD 5C9A
uid                               Boris Samorodov (FreeBSD) <bsam@freebsd.org>
sub 2048g/7753A3F1 2006-06-21
```

D.3.275. Mark Santcroos <marks@FreeBSD.org>

```
pub 1024D/DBE7EB8E 2005-03-08
    Key fingerprint = C0F0 44F3 3F15 520F 6E32 186B BE0A BA42 DBE7 EB8E
uid                               Mark Santcroos <marks@ripe.net>
uid                               Mark Santcroos <mark@santcroos.net>
uid                               Mark Santcroos <marks@freebsd.org>
sub 2048g/FFF80F85 2005-03-08
```

D.3.276. Bernhard Schmidt <bschmidt@FreeBSD.org>

```
pub 1024D/5F754FBC 2009-06-15
    Key fingerprint = 6B87 C8A9 6BA5 6B18 11CF 8C38 A1B7 0731 5F75 4FBC
uid                               Bernhard Schmidt <bschmidt@FreeBSD.org>
uid                               Bernhard Schmidt <bschmidt@techwires.net>
sub 1024g/1945DC1D 2009-06-15
```

D.3.277. Wolfram Schneider <wosch@FreeBSD.org>

```
Type Bits/KeyID      Date      User ID
pub 1024/2B7181AD 1997/08/09 Wolfram Schneider <wosch@FreeBSD.org>
    Key fingerprint = CA 16 91 D9 75 33 F1 07 1B F0 B4 9F 3E 95 B6 09
```

D.3.278. Ed Schouten <ed@FreeBSD.org>

```
pub 4096R/3491A2BB 2011-03-12 [expires: 2016-03-10]
    Key fingerprint = A110 5982 A887 74A2 F4B1 D70A 6E5E D8FE 3491 A2BB
uid                               Ed Schouten (The FreeBSD Project) <ed@FreeBSD.org>
uid                               Ed Schouten <ed@80386.nl>
sub 4096R/81BB41E6 2011-03-12 [expires: 2016-03-10]
```

D.3.279. David Schultz <das@FreeBSD.org>

```
pub 1024D/BE848B57 2001-07-19 David Schultz <das@FreeBSD.ORG>
    Key fingerprint = 0C12 797B A9CB 19D9 FDAF 2A39 2D76 A2DB BE84 8B57
uid David Schultz <dschultz@uclink.Berkeley.EDU>
uid David Schultz <das@FreeBSD.ORG>
sub 2048g/69206E8E 2001-07-19
```

D.3.280. Michael Scheidell <scheidell@FreeBSD.org>

```
pub 2048R/34622C1D 2011-11-16
    Key fingerprint = 0A0C 9ECA 18EC 47AC C715 2187 91B9 F9FE 3462 2C1D
uid                               Michael Scheidell <scheidell@freebsd.org>
```

sub 2048R/8F241971 2011-11-16

D.3.281. Jens Schweikhardt <schweikh@FreeBSD.org>

```
pub 1024D/0FF231FD 2002-01-27 Jens Schweikhardt <schweikh@FreeBSD.org>
   Key fingerprint = 3F35 E705 F02F 35A1 A23E 330E 16FE EA33 0FF2 31FD
uid                               Jens Schweikhardt <schweikh@schweikhardt.net>
sub 1024g/6E93CACC 2002-01-27 [expires: 2005-01-26]
```

D.3.282. Matthew Seaman <matthew@FreeBSD.org>

```
pub 1024D/60AE908C 2005-12-17 [expires: 2012-03-21]
   Key fingerprint = B555 2A96 274E D248 5734 0EB4 F0C8 E4E7 60AE 908C
uid                               Matthew Seaman <m.seaman@infracaninophile.co.uk>
uid                               Matthew Seaman <m.seaman@black-earth.co.uk>
uid                               Matthew Seaman <matthew@freebsd.org>
sub 2048g/58BFDA29 2005-12-17 [expires: 2012-03-21]
sub 1024D/9B19F956 2006-12-18 [expires: 2012-03-21]
```

D.3.283. Stanislav Sedov <stas@FreeBSD.org>

```
pub 4096R/092FD9F0 2009-05-23
   Key fingerprint = B83A B15D 929A 364A D8BC B3F9 BF25 A231 092F D9F0
uid                               Stanislav Sedov <stas@FreeBSD.org>
uid                               Stanislav Sedov <stas@SpringDaemons.com>
uid                               Stanislav Sedov (Corporate email) <stas@deglitch.com>
uid                               Stanislav Sedov (Corporate email) <stas@ht-systems.ru>
uid                               Stanislav Sedov (Corporate email) <ssedov@3playnet.com>
uid                               Stanislav Sedov <ssedov@mbsd.msk.ru>
uid                               Stanislav Sedov (Corporate email) <ssedov@swifttest.com>
sub 4096R/6FD2025F 2009-05-23
```

D.3.284. Johan van Selst <johans@FreeBSD.org>

```
pub 4096R/D3AE8D3A 2009-09-01
   Key fingerprint = 31C8 D089 DDB6 96C6 F3C1 29C0 A9C8 6C8D D3AE 8D3A
uid                               Johan van Selst
uid                               Johan van Selst <johans@gletsjer.net>
uid                               Johan van Selst <johans@stack.nl>
uid                               Johan van Selst <johans@FreeBSD.org>
uid                               Johan van Selst (GSWoT:NL50) <johans@gswot.org>
sub 2048R/B002E38C 2009-09-01
sub 2048R/1EBCAECB 2009-09-01
sub 2048R/639A1446 2009-09-01
sub 3072D/6F2708F4 2009-09-01
sub 4096g/D6F89E83 2009-09-01
```

D.3.285. Bakul Shah <bakul@FreeBSD.org>

```
pub 1024D/86AEE4CB 2006-04-20
    Key fingerprint = 0389 26E8 381C 6980 AEC0 10A5 E540 A157 86AE E4CB
uid      Bakul Shah <bakul@freebsd.org>
sub 2048g/5C3DCC24 2006-04-20
```

D.3.286. Gregory Neil Shapiro <gshapiro@FreeBSD.org>

```
pub 1024R/4FBE2ADD 2000-10-13 Gregory Neil Shapiro <gshapiro@gshapiro.net>
    Key fingerprint = 56 D5 FF A7 A6 54 A6 B5 59 10 00 B9 5F 5F 20 09
uid      Gregory Neil Shapiro <gshapiro@FreeBSD.org>

pub 1024D/F76A9BF5 2001-11-14 Gregory Neil Shapiro <gshapiro@FreeBSD.org>
    Key fingerprint = 3B5E DAF1 4B04 97BA EE20 F841 21F9 C5BC F76A 9BF5
uid      Gregory Neil Shapiro <gshapiro@gshapiro.net>
sub 2048g/935657DC 2001-11-14

pub 1024D/FCE56561 2000-10-14 Gregory Neil Shapiro <gshapiro@FreeBSD.org>
    Key fingerprint = 42C4 A87A FD85 C34F E77F 5EA1 88E1 7B1D FCE5 6561
uid      Gregory Neil Shapiro <gshapiro@gshapiro.net>
sub 1024g/285DC8A0 2000-10-14 [expires: 2001-10-14]
```

D.3.287. Arun Sharma <arun@FreeBSD.org>

```
pub 1024D/7D112181 2003-03-06 Arun Sharma <arun@sharma-home.net>
    Key fingerprint = A074 41D6 8537 C7D5 070E 0F78 0247 1AE2 7D11 2181
uid      Arun Sharma <arun@freebsd.org>
uid      Arun Sharma <arun.sharma@intel.com>
sub 1024g/ACAD98DA 2003-03-06 [expires: 2005-03-05]
```

D.3.288. Wesley Shields <wxs@FreeBSD.org>

```
pub 1024D/17F0AA37 2007-12-27
    Key fingerprint = 96D1 2E6B F61C 2F3D 83EF 8F0B BE54 310C 17F0 AA37
uid      Wesley Shields <wxs@FreeBSD.org>
uid      Wesley Shields <wxs@atarininja.org>
sub 2048g/2EDA1BB8 2007-12-27
```

D.3.289. Norikatsu Shigemura <nork@FreeBSD.org>

```
pub 1024D/7104EA4E 2005-02-14
    Key fingerprint = 9580 60A3 B58A 0864 79CB 779A 6FAE 229B 7104 EA4E
uid      Norikatsu Shigemura <nork@cityfujisawa.ne.jp>
uid      Norikatsu Shigemura <nork@ninth-nine.com>
uid      Norikatsu Shigemura <nork@FreeBSD.org>
```

sub 4096g/EF56997E 2005-02-14

D.3.290. Shteryana Shopova <syrinx@FreeBSD.org>

pub 1024D/1C139BC5 2006-10-07
 Key fingerprint = B83D 2451 27AB B767 504F CB85 4FB1 C88B 1C13 9BC5
 uid Shteryana Shopova (syrinx) <shteryana@FreeBSD.org>
 sub 2048g/6D2E9C98 2006-10-07

D.3.291. Vanilla I. Shu <vanilla@FreeBSD.org>

pub 1024D/ACE75853 2001-11-20 Vanilla I. Shu <vanilla@FreeBSD.org>
 Key fingerprint = 290F 9DB8 42A3 6257 5D9A 5585 B25A 909E ACE7 5853
 sub 1024g/CE695D0E 2001-11-20

D.3.292. Ashish SHUKLA <ashish@FreeBSD.org>

pub 4096R/E74FA4B0 2010-04-13
 Key fingerprint = F682 CDCC 39DC 0FEA E116 20B6 C746 CFA9 E74F A4B0
 uid Ashish SHUKLA <wahjava@gmail.com>
 uid Ashish SHUKLA <wahjava@googlemail.com>
 uid Ashish SHUKLA <wahjava.ml@gmail.com>
 uid Ashish SHUKLA <wahjava@members.fsf.org>
 uid Ashish SHUKLA <wahjava@perl.org.in>
 uid Ashish SHUKLA <wahjava@users.sourceforge.net>
 uid Ashish SHUKLA <wah.java@yahoo.com>
 uid Ashish SHUKLA <wah_java@hotmail.com>
 uid Ashish SHUKLA <ashish.shukla@airtelmail.in>
 uid Ashish SHUKLA <wahjava@member.fsf.org>
 uid [jpeg image of size 4655]
 uid Ashish SHUKLA (FreeBSD Committer Address) <ashish@FreeBSD.ORG>
 sub 4096R/F20D202D 2010-04-13

D.3.293. Bruce M. Simpson <bms@FreeBSD.org>

pub 1024D/860DB53B 2003-08-06 Bruce M Simpson <bms@freebsd.org>
 Key fingerprint = 0D5F 1571 44DF 51B7 8B12 041E B9E5 2901 860D B53B
 sub 2048g/A2A32D8B 2003-08-06 [expires: 2006-08-05]

D.3.294. Dmitry Sivachenko <demon@FreeBSD.org>

pub 1024D/13D5DF80 2002-03-18 Dmitry Sivachenko <mitya@cavia.pp.ru>
 Key fingerprint = 72A9 12C9 BB02 46D4 4B13 E5FE 1194 9963 13D5 DF80
 uid Dmitry S. Sivachenko <demon@FreeBSD.org>

sub 1024g/060F6DBD 2002-03-18

D.3.295. Jesper Skriver <jesper@FreeBSD.org>

```
pub 1024D/F9561C31 2001-03-09 Jesper Skriver <jesper@FreeBSD.org>
    Key fingerprint = 6B88 9CE8 66E9 E631 C9C5 5EB4 22AB F0EC F956 1C31
uid                                     Jesper Skriver <jesper@skriver.dk>
uid                                     Jesper Skriver <jesper@wheel.dk>
sub 1024g/777C378C 2001-03-09
```

D.3.296. Ville Skyttä <scop@FreeBSD.org>

```
pub 1024D/BCD241CB 2002-04-07 Ville Skyttä <ville.skytta@iki.fi>
    Key fingerprint = 4E0D EBAB 3106 F1FA 3FA9 B875 D98C D635 BCD2 41CB
uid                                     Ville Skyttä <ville.skytta@xemacs.org>
uid                                     Ville Skyttä <scop@FreeBSD.org>
sub 2048g/9426F4D1 2002-04-07
```

D.3.297. Andrey Slusar <anray@FreeBSD.org>

```
pub 1024D/AE7B5418 2005-12-12
    Key fingerprint = DE70 C24B 55A0 4A06 68A1 D425 3C59 9A9B AE7B 5418
uid                                     Andrey Slusar <anray@ext.by>
uid                                     Andrey Slusar <anrays@gmail.com>
uid                                     Andrey Slusar <anray@FreeBSD.org>
sub 2048g/7D0EB77D 2005-12-12
```

D.3.298. Florian Smeets <flo@FreeBSD.org>

```
pub 1024D/C942BF09 2008-10-24
    Key fingerprint = 54BB 157B 8DB2 9E46 4A3C 69AB 6A9A 3C3F C942 BF09
uid                                     Florian Smeets <flo@smeets.im>
uid                                     Florian Smeets <flo@kasimir.com>
uid                                     Florian Smeets <flo@FreeBSD.org>
sub 2048g/4AAF040E 2008-10-24
```

D.3.299. Gleb Smirnoff <glebius@FreeBSD.org>

```
pub 1024D/1949DC80 2003-08-25
    Key fingerprint = 872C E14A 2F03 A3E8 D882 026E 5DE4 D7FE 1949 DC80
uid                                     Gleb Smirnoff <glebius@FreeBSD.org>
uid                                     Gleb Smirnoff <glebius@cell.sick.ru>
uid                                     Gleb Smirnoff <glebius@bestcom.ru>
uid                                     Gleb Smirnoff <glebius@rambler-co.ru>
```

```
uid          Gleb Smirnoff <glebius@freebsd.org>
uid          Gleb Smirnoff <glebius@freebsd.int.ru>
sub 1024g/A05118BD 2003-08-25
```

D.3.300. Ken Smith <kensmith@FreeBSD.org>

```
pub 1024D/29AEA7F6 2003-12-02 Ken Smith <kensmith@cse.buffalo.edu>
   Key fingerprint = 4AB7 D302 0753 8215 31E7 F1AD FC6D 7855 29AE A7F6
uid          Ken Smith <kensmith@freebsd.org>
sub 1024g/0D509C6C 2003-12-02
```

D.3.301. Ben Smithurst <ben@FreeBSD.org>

```
pub 1024D/2CEF442C 2001-07-11 Ben Smithurst <ben@LSRfm.com>
   Key fingerprint = 355D 0FFF B83A 90A9 D648 E409 6CFC C9FB 2CEF 442C
uid          Ben Smithurst <ben@vinosystems.com>
uid          Ben Smithurst <ben@smithurst.org>
uid          Ben Smithurst <ben@FreeBSD.org>
uid          Ben Smithurst <csxbsc@comp.leeds.ac.uk>
uid          Ben Smithurst <ben@scientia.demon.co.uk>
sub 1024g/347071FF 2001-07-11
```

D.3.302. Dag-Erling C. Smørgrav <des@FreeBSD.org>

```
pub 1024D/64EBE220 2006-11-11 [expires: 2012-12-31]
   Key fingerprint = 3A1C 8E68 952C 3305 6984 6486 30D4 3A6E 64EB E220
uid          Dag-Erling Smørgrav <des@des.no>
uid          Dag-Erling Smørgrav <des@freebsd.org>
uid          Dag-Erling Smørgrav <des@usit.uio.no>
uid          [jpeg image of size 3315]
sub 2048g/920C3313 2006-11-11 [expires: 2012-12-31]
```

D.3.303. Maxim Sobolev <sobomax@FreeBSD.org>

```
pub 1024D/888205AF 2001-11-21 Maxim Sobolev <sobomax@FreeBSD.org>
   Key fingerprint = 85C9 DCB0 6828 087C C977 3034 A0DB B9B7 8882 05AF
uid          Maxim Sobolev <sobomax@mail.ru>
uid          Maxim Sobolev <sobomax@altavista.net>
uid          Maxim Sobolev <vegacap@i.com.ua>

pub 1024D/468EE6D8 2003-03-21 Maxim Sobolev <sobomax@portaone.com>
   Key fingerprint = 711B D315 3360 A58F 9A0E 89DB 6D40 2558 468E E6D8
uid          Maxim Sobolev <sobomax@FreeBSD.org>
uid          Maxim Sobolev <sobomax@mail.ru>
uid          Maxim Sobolev <vegacap@i.com.ua>
```

```
pub 1024D/6BEC980A 2004-02-13 Maxim Sobolev <sobomax@portaone.com>
    Key fingerprint = 09D5 47B4 8D23 626F B643 76EB DFEE 3794 6BEC 980A
uid Maxim Sobolev <sobomax@FreeBSD.org>
uid Maksym Sobolyev (It's how they call me in official documents. Pret
uid Maksym Sobolyev (It's how they call me in official documents. Pret
sub 2048g/16D049AB 2004-02-13 [expires: 2005-02-12]
```

D.3.304. Brian Somers <brian@FreeBSD.org>

```
pub 1024R/666A7421 1997-04-30 Brian Somers <brian@freebsd-services.com>
    Key fingerprint = 2D 91 BD C2 94 2C 46 8F 8F 09 C4 FC AD 12 3B 21
uid Brian Somers <brian@awfulhak.org>
uid Brian Somers <brian@FreeBSD.org>
uid Brian Somers <brian@OpenBSD.org>
uid Brian Somers <brian@uk.FreeBSD.org>
uid Brian Somers <brian@uk.OpenBSD.org>
```

D.3.305. Stacey Son <:sson@FreeBSD.org>

```
pub 1024D/CE8319F3 2008-07-08
    Key fingerprint = 64C7 8D92 C1DF B940 1171 5ED3 186A 758A CE83 19F3
uid Stacey Son <:sson@FreeBSD.org>
uid Stacey Son <stacey@son.org>
uid Stacey Son <:sson@byu.net>
uid Stacey Son <:sson@secure.net>
uid Stacey Son <:sson@dev-random.com>
sub 2048g/0F724E52 2008-07-08
```

D.3.306. Nicolas Souchu <nsouch@FreeBSD.org>

```
pub 1024D/C744F18B 2002-02-13 Nicholas Souchu <nsouch@freebsd.org>
    Key fingerprint = 992A 144F AC0F 40BA 55AE DE6D 752D 0A6C C744 F18B
sub 1024g/90BD3231 2002-02-13
```

D.3.307. Suleiman Souhlal <ssouhlal@FreeBSD.org>

```
pub 1024D/2EA50469 2004-07-24 Suleiman Souhlal <ssouhlal@FreeBSD.org>
    Key fingerprint = DACF 89DB 54C7 DA1D 37AF 9A94 EB55 E272 2EA5 0469
sub 2048g/0CDCC535 2004-07-24
```

D.3.308. Ulrich Spörlein <uqs@FreeBSD.org>

```
pub 2048R/4AAF82CE 2010-01-27 [expires: 2015-01-26]
    Key fingerprint = 08DF A6A0 B1EB 98A5 EDDA 9005 A3A6 9864 4AAF 82CE
uid      Ulrich Spörlein <uqs@spoerlein.net>
uid      Ulrich Spoerlein <uspoerlein@gmail.com>
uid      Ulrich Spörlein (The FreeBSD Project) <uqs@FreeBSD.org>
uid      Ulrich Spörlein <ulrich.spoerlein@web.de>
sub 2048R/162E8BD2 2010-01-27 [expires: 2015-01-26]
```

D.3.309. Rink Springer <rink@FreeBSD.org>

```
pub 1024D/ECEDBFFF 2003-09-19
    Key fingerprint = A8BE 9C82 9B81 4289 A905 418D 6F73 BAD2 ECED BFFF
uid      Rink Springer <rink@il.fontys.nl>
uid      Rink Springer (FreeBSD Project) <rink@FreeBSD.org>
uid      Rink Springer <rink@stack.nl>
sub 2048g/3BC3E67E 2003-09-19
```

D.3.310. Vsevolod Stakhov <vsevolod@FreeBSD.org>

```
pub 4096R/90081437 2012-05-16 [expires: 2017-05-15]
    Key fingerprint = DD9A 126C E675 1EA5 2A97 04A3 0764 7B67 9008 1437
uid      Vsevolod Stakhov <vsevolod@FreeBSD.org>
sub 4096R/4A5A0B54 2012-05-16 [expires: 2017-05-15]
```

D.3.311. Ryan Steinmetz <zi@FreeBSD.org>

```
pub 1024D/7AD7FAF2 2004-01-21
    Key fingerprint = EF36 D45A 5CA9 28B1 A550 18CD A43C D111 7AD7 FAF2
uid      Ryan Steinmetz <zi@FreeBSD.org>
uid      Ryan Steinmetz <rpsfa@rit.edu>
uid      Ryan Steinmetz <zi@zi0r.com>
sub 1024g/058BC057 2004-01-21
sub 4096g/0EB108D2 2006-02-27
sub 1024D/FEF36DD7 2006-02-27
```

D.3.312. Randall R. Stewart <rrs@FreeBSD.org>

```
pub 1024D/0373B8B2 2006-09-01
    Key fingerprint = 74A6 810E 6DEA D69B 6496 5FA9 8AEF 4166 0373 B8B2
uid      Randall R Stewart <randall@lakerest.net>
uid      Randall R Stewart <rrs@cisco.com>
uid      Randall R Stewart <rrs@FreeBSD.org>
sub 2048g/88027C0B 2006-09-01
```

D.3.313. Murray Stokely <murray@FreeBSD.org>

```
pub 1024D/0E451F7D 2001-02-12 Murray Stokely <murray@freebsd.org>
    Key fingerprint = E2CA 411D DD44 53FD BB4B 3CB5 B4D7 10A2 0E45 1F7D
sub 1024g/965A770C 2001-02-12
```

D.3.314. Volker Stolz <vs@FreeBSD.org>

```
pub 1024R/3FD1B6B5 1998-06-16 Volker Stolz <vs@freebsd.org>
    Key fingerprint = 69 6F BD A0 2E FE 19 66 CF B9 68 6E 41 7D F9 B9
uid                               Volker Stolz <stolz@i2.informatik.rwth-aachen.de> (LSK)
uid                               Volker Stolz <vs@foldr.org>
```

D.3.315. Ryan Stone <rstone@FreeBSD.org>

```
pub 1024D/3141B73A 2010-04-13
    Key fingerprint = 4A6D DC04 DDC5 0822 2687 A086 FD3F 16CB 3141 B73A
uid                               Ryan Stone (FreeBSD) <rstone@freebsd.org>
sub 2048g/A8500B5F 2010-04-13
```

D.3.316. Søren Straarup <xride@FreeBSD.org>

```
pub 1024D/E683AD40 2006-09-28
    Key fingerprint = 8A0E 7E57 144B BC25 24A9 EC1A 0DBC 3408 E683 AD40
uid                               Soeren Straarup <xride@xride.dk>
uid                               Soeren Straarup <xride@FreeBSD.org>
uid                               Soeren Straarup <xride@x12.dk>
sub 2048g/2B18B3B8 2006-09-28
```

D.3.317. Marius Strobl <marius@FreeBSD.org>

```
pub 1024D/E0AC6F8D 2004-04-16
    Key fingerprint = 3A6C 4FB1 8BB9 4F2E BDDC 4AB6 D035 799C E0AC 6F8D
uid                               Marius Strobl <marius@FreeBSD.org>
uid                               Marius Strobl <marius@alchemy.franken.de>
sub 1024g/08BBD875 2004-04-16
```

D.3.318. Carlo Strub <cs@FreeBSD.org>

```
pub 4096R/48C0D3F8 2012-06-05 [expires: 2017-06-04]
    Key fingerprint = A2DD 6235 93E4 6520 1AF9 2B14 E925 8207 48C0 D3F8
uid                               Carlo Strub <cs@FreeBSD.org>
sub 4096R/99AC2A67 2012-06-05 [expires: 2017-06-04]
```

```
pub 4096R/2C23E493 2012-06-08 [expires: 2017-06-07]
    Key fingerprint = 26E0 89EA B9BE 5FC5 D740 CBB4 EABC 7343 2C23 E493
uid          Carlo Strub (Mail) <cs@carlostrub.ch>
sub 4096R/DF86FD3E 2012-06-08 [expires: 2017-06-07]
```

D.3.319. Cheng-Lung Sung <clsung@FreeBSD.org>

```
pub 1024D/956E8BC1 2003-09-12 Cheng-Lung Sung <clsung@FreeBSD.org>
    Key fingerprint = E0BC 57F9 F44B 46C6 DB53 8462 F807 89F3 956E 8BC1
uid          Cheng-Lung Sung (Software Engineer) <clsung@dragon2.net>
uid          Cheng-Lung Sung (Alumnus of CSIE, NCTU, Taiwan) <clsung@sungsung.c
uid          Cheng-Lung Sung (AlanSung) <clsung@tiger2.net>
uid          Cheng-Lung Sung (FreeBSD@Taiwan) <clsung@freebsd.csie.nctu.edu.tw>
uid          Cheng-Lung Sung (Ph.D. Student of NTU.EECS) <d92921016@ntu.edu.tw>
uid          Cheng-Lung Sung (FreeBSD Freshman) <clsung@tw.freebsd.org>
uid          Cheng-Lung Sung (ports committer) <clsung@FreeBSD.org>
sub 1024g/1FB800C2 2003-09-12
```

D.3.320. Gregory Sutter <gsutter@FreeBSD.org>

```
pub 1024D/845DFEDD 2000-10-10 Gregory S. Sutter <gsutter@zer0.org>
    Key fingerprint = D161 E4EA 4BFA 2427 F3F9 5B1F 2015 31D5 845D FEDD
uid          Gregory S. Sutter <gsutter@freebsd.org>
uid          Gregory S. Sutter <gsutter@daemonnews.org>
uid          Gregory S. Sutter <gsutter@pobox.com>
sub 2048g/0A37BBCE 2000-10-10
```

D.3.321. Koichi Suzuki <metal@FreeBSD.org>

```
pub 1024D/AE562682 2004-05-23 SUZUKI Koichi <metal@FreeBSD.org>
    Key fingerprint = 92B9 A202 B5AB 8CB6 89FC 6DD1 5737 C702 AE56 2682
sub 4096g/730E604B 2004-05-23
```

D.3.322. Ryusuke SUZUKI <ryusuke@FreeBSD.org>

```
pub 1024D/63D29724 2009-12-18
    Key fingerprint = B108 7109 2E62 BECB 0F78 FE65 1B9A D1BE 63D2 9724
uid          Ryusuke SUZUKI <ryusuke@FreeBSD.org>
uid          Ryusuke SUZUKI <ryusuke@jp.FreeBSD.org>
sub 1024g/5E4DD044 2009-12-18
```

D.3.323. Gary W. Swearingen <garys@FreeBSD.org>

```
pub 1024D/FAA48AD5 2005-08-22 [expires: 2007-08-22]
    Key fingerprint = 8292 CC3E 81B5 E54F E3DD F987 FA52 E643 FAA4 8AD5
uid Gary W. Swearingen <garys@freebsd.org>
sub 2048g/E34C3CA0 2005-08-22 [expires: 2007-08-22]
```

D.3.324. Yoshihiro Takahashi <nyan@FreeBSD.org>

```
pub 1024D/8394B81F 2001-10-15 Yoshihiro TAKAHASHI <nyan@jp.FreeBSD.org>
    Key fingerprint = D4FA D8CA 2AED FCF4 90A3 3569 8666 0500 8394 B81F
uid Yoshihiro TAKAHASHI <nyan@furiru.org>
uid Yoshihiro TAKAHASHI <nyan@FreeBSD.org>
sub 1024g/B796F020 2001-10-15
```

D.3.325. Sahil Tandon <sahil@FreeBSD.org>

```
pub 2048R/C016D977 2010-04-08
    Key fingerprint = 6AD2 BA99 8E3A 8DA6 DFC1 53CF DBD0 6001 C016 D977
uid Sahil Tandon <sahil@tandon.net>
uid Sahil Tandon <sahil@FreeBSD.org>
sub 2048R/F7776FBC 2010-04-08
```

D.3.326. TAKATSU Tomonari <tota@FreeBSD.org>

```
pub 1024D/67F58F29 2009-05-17
    Key fingerprint = 6940 B575 FC4A FA26 C094 279A 4B9B 6326 67F5 8F29
uid TAKATSU Tomonari <tota@FreeBSD.org>
sub 2048g/18B112CD 2009-05-17
```

D.3.327. Romain Tartière <romain@FreeBSD.org>

```
pub 3072R/5112336F 2010-04-09
    Key fingerprint = 8234 9A78 E7C0 B807 0B59 80FF BA4D 1D95 5112 336F
uid Romain Tartière <romain@blogreen.org>
uid Romain Tartière (FreeBSD) <romain@FreeBSD.org>
sub 3072R/C1B2B656 2010-04-09
sub 3072R/8F8125F4 2010-04-09
```

D.3.328. Sylvio Cesar Teixeira <sylvio@FreeBSD.org>

```
pub 2048R/AA7395A1 2009-10-28
    Key fingerprint = B319 6AAF 0016 4308 6D93 E652 3C5F 21A2 AA73 95A1
uid Sylvio Cesar Teixeira (My key) <sylvio@FreeBSD.org>
```

sub 2048R/F758F556 2009-10-28

D.3.329. Ion-Mihai Tetcu <itetcu@FreeBSD.org>

```
pub 1024D/21FFA1E5 2008-05-08 [expires: 2010-05-08]
    Key fingerprint = A880 42DD BD71 BAA5 AED7 AEA2 27B1 88BA 21FF A1E5
uid      Ion-Mihai "IOnut" Tetcu <itetcu@FreeBSD.org>
sub 2048g/0B30E680 2008-05-08 [expires: 2010-05-08]
```

D.3.330. Mikhail Teterin <mi@FreeBSD.org>

```
pub 1024R/3FC71479 1995-09-08 Mikhail Teterin <mi@aldan.star89.galstar.com>
    Key fingerprint = 5F 15 EA 78 A5 40 6A 0F 14 D7 D9 EA 6E 2B DA A4
```

D.3.331. Gordon Tetlow <gordon@FreeBSD.org>

```
pub 1024D/357D65FB 2002-05-14 Gordon Tetlow <gordont@gnf.org>
    Key fingerprint = 34EF AD12 10AF 560E C3AE CE55 46ED ADF4 357D 65FB
uid      Gordon Tetlow <gordon@FreeBSD.org>
sub 1024g/243694AB 2002-05-14
```

D.3.332. Lars Thegler <lth@FreeBSD.org>

```
pub 1024D/56B0CA08 2004-05-31 Lars Thegler <lth@FreeBSD.org>
    Key fingerprint = ABAB F98C EA78 1C8D 6FDD CB27 1CA9 5A63 56B0 CA08
uid      Lars Thegler <lars@thegler.dk>
sub 1024g/E8C58EF3 2004-05-31
```

D.3.333. Jase Thew <jase@FreeBSD.org>

```
pub 3072R/3EEAF1EB 2012-05-30
    Key fingerprint = F5FB 959F CF1B 6550 054E 2819 A484 BCDB 3EEA F1EB
uid      Jase Thew (FreeBSD) <jase@FreeBSD.org>
uid      Jase Thew <freebsd@beardz.net>
```

D.3.334. David Thiel <lx@FreeBSD.org>

```
pub 1024D/A887A9B4 2006-11-30 [expires: 2011-11-29]
    Key fingerprint = F08F 6A12 738F C9DF 51AC 8C62 1E30 7CBE A887 A9B4
uid      David Thiel <lx@FreeBSD.org>
sub 2048g/B9BD92C5 2006-11-30 [expires: 2011-11-29]
```

D.3.335. Fabien Thomas <fabient@FreeBSD.org>

```
pub 1024D/07745930 2009-03-16
    Key fingerprint = D8AC EFA2 2FBD 7788 9628 4E8D 3F35 3B88 0774 5930
uid Fabien Thomas <fabient@FreeBSD.org>
sub 2048g/BC173395 2009-03-16
```

D.3.336. Thierry Thomas <thierry@FreeBSD.org>

```
pub 1024D/C71405A2 1997-10-11
    Key fingerprint = 3BB8 F358 C2F1 776C 65C9 AE51 73DE 698C C714 05A2
uid Thierry Thomas <thierry@pompo.net>
uid Thierry Thomas <tthomas@mail.dotcom.fr>
uid Thierry Thomas (FreeBSD committer) <thierry@FreeBSD.org>
sub 1024R/C5529925 2003-11-26
sub 2048g/05CF3992 2008-02-05
```

D.3.337. Andrew Thompson <thompsa@FreeBSD.org>

```
pub 1024D/BC6B839B 2005-05-05
    Key fingerprint = DE74 3F49 B97C A170 C8F1 8423 CAB6 9D57 BC6B 839B
uid Andrew Thompson <thompsa@freebsd.org>
uid Andrew Thompson <andy@fud.org.nz>
sub 2048g/92E370FB 2005-05-05
```

D.3.338. Florent Thoumie <flz@FreeBSD.org>

```
pub 1024D/5147DCF4 2004-12-04
    Key fingerprint = D203 AF5F F31A 63E2 BFD5 742B 3311 246D 5147 DCF4
uid Florent Thoumie (FreeBSD committer address) <flz@FreeBSD.org>
uid Florent Thoumie (flz) <florent@thoumie.net>
uid Florent Thoumie (flz) <flz@xbsd.org>
uid [jpeg image of size 1796]
sub 2048g/15D930B9 2004-12-04
```

D.3.339. Yar Tikhyy <yar@FreeBSD.org>

```
pub 1024D/EA04CF5A 2008-08-31
    Key fingerprint = C063 6788 AFF2 A62F 06B7 516D 200F 06AF EA04 CF5A
uid Yar Tikhyy <yar@freebsd.org>
sub 2048g/20443F06 2008-08-31
```

D.3.340. Jilles Tjoelker <jilles@FreeBSD.org>

```
pub 4096R/D5AE6220 2011-07-02
    Key fingerprint = 4AF5 F1CC BDD7 700B F005 79A4 A2C4 C4D4 D5AE 6220
uid Jilles Tjoelker <jilles@stack.nl>
uid Jilles Tjoelker <tjoelker@zonnet.nl>
uid Jilles Tjoelker (FreeBSD) <jilles@FreeBSD.org>
sub 4096R/14CB5775 2011-07-02
```

D.3.341. Ganbold Tsagaankhuu <ganbold@FreeBSD.org>

```
pub 1024D/78F6425E 2008-02-26 [expires: 2013-02-24]
    Key fingerprint = 9B8E DC41 D3F4 F7FC D8EA 417C D4F7 2AEF 78F6 425E
uid Ganbold <ganbold@freebsd.org>
sub 2048g/716FCBF9 2008-02-26 [expires: 2013-02-24]
```

D.3.342. Michael Tuexen <tuexen@FreeBSD.org>

```
pub 1024D/04EEDABE 2009-06-08
    Key fingerprint = 493A CCB8 60E6 5510 A01D 360E 8497 B854 04EE DABE
uid Michael Tuexen <tuexen@FreeBSD.org>
sub 2048g/F653AA03 2009-06-08
```

D.3.343. Andrew Turner <andrew@FreeBSD.org>

```
pub 2048R/31B31614 2010-07-01
    Key fingerprint = 08AC 2C57 F14F FDD1 2232 B5CD AA16 EFB8 31B3 1614
uid Andrew Turner <andrew@freebsd.org>
uid Andrew Turner <andrew@fubar.geek.nz>
sub 2048R/9ACBF138 2010-07-01
```

D.3.344. Hajimu UMEMOTO <ume@FreeBSD.org>

```
pub 1024D/BF9071FE 2005-03-17
    Key fingerprint = 1F00 0B9E 2164 70FC 6DC5 BF5F 04E9 F086 BF90 71FE
uid Hajimu UMEMOTO <ume@mahoroba.org>
uid Hajimu UMEMOTO <ume@FreeBSD.org>
uid Hajimu UMEMOTO <ume@jp.FreeBSD.org>
sub 2048g/748DB3B0 2005-03-17
```

D.3.345. Stephan Uphoff <ups@FreeBSD.org>

```
pub 2048R/D684B04A 2004-10-06 Stephan Uphoff <ups@freebsd.org>
   Key fingerprint = B5D2 04AE CA8F 7055 7474 3C85 F908 7F55 D684 B04A
uid                               Stephan Uphoff <ups@tree.com>
sub 2048R/A15F921B 2004-10-06
```

D.3.346. Jacques Vidrine <nectar@FreeBSD.org>

```
pub 2048R/33C1627B 2001-07-05 Jacques A. Vidrine <nectar@celabo.org>
   Key fingerprint = CB CE 7D A0 6E 01 DC 61 E5 91 0A BE 79 17 D3 82
uid                               Jacques A. Vidrine <jvidrine@verio.net>
uid                               Jacques A. Vidrine <n@nectar.com>
uid                               Jacques A. Vidrine <jacques@vidrine.cc>
uid                               Jacques A. Vidrine <nectar@FreeBSD.org>
uid                               Jacques A. Vidrine <n@nectar.cc>

pub 1024D/1606DB95 2001-07-05 Jacques A. Vidrine <nectar@celabo.org>
   Key fingerprint = 46BC EA5B F70A CC81 5332 0832 8C32 8CFF 1606 DB95
uid                               Jacques A. Vidrine <jvidrine@verio.net>
uid                               Jacques A. Vidrine <n@nectar.com>
uid                               Jacques A. Vidrine <jacques@vidrine.cc>
uid                               Jacques A. Vidrine <nectar@FreeBSD.org>
uid                               Jacques A. Vidrine <n@nectar.cc>
sub 2048g/57EDEA6F 2001-07-05
```

D.3.347. Alberto Villa <avilla@FreeBSD.org>

```
pub 1024R/44350A8B 2010-01-24
   Key fingerprint = F740 CE4E EDDD DA9B 4A1B 1445 DF18 82EA 4435 0A8B
uid                               Alberto Villa <avilla@FreeBSD.org>
sub 1024R/F7C8254C 2010-01-24
```

D.3.348. Nicola Vitale <nivit@FreeBSD.org>

```
pub 1024D/F11699E5 2006-12-05
   Key fingerprint = 2C17 C591 2C6D 82BD F3DB F1BF 8FC9 6763 F116 99E5
uid                               Nicola Vitale (Public key for nivit@FreeBSD.org) <nivit@FreeBSD.org>
sub 2048g/4C90805D 2006-12-05
```

D.3.349. Ivan Voras <ivoras@FreeBSD.org>

```
pub 1024D/569C05C8 2000-05-24
   Key fingerprint = AB9A A555 C47C B61D BF83 154C 95D9 C041 569C 05C8
uid                               Ivan Voras <ivoras@fer.hr>
uid                               Ivan Voras <ivan.voras@fer.hr>
```

```
uid          Ivan Voras <ivoras@geri.cc.fer.hr>
uid          [jpeg image of size 4567]
uid          Ivan Voras <ivoras@sharanet.org>
uid          Ivan Voras <ivoras@gmail.com>
uid          Ivan Voras <ivoras@yahoo.com>
uid          Ivan Voras <ivoras@freebsd.org>
uid          Ivan Voras <ivan.voras@zg.t-com.hr>
sub 1536g/149FDD60 2000-05-24
```

D.3.350. Stefan Walter <stefan@FreeBSD.org>

```
pub 3072R/12B9E0B3 2003-03-06
   Key fingerprint = 85D8 6A49 22C7 6CD9 B011 5D6A 5691 111B 12B9 E0B3
uid          Stefan Walter <stefan@freebsd.org>
uid          Stefan Walter <sw@gegenunendlich.de>
sub 3072R/6D35457A 2003-03-06
```

D.3.351. Kai Wang <kaiw@FreeBSD.org>

```
pub 1024D/AEB910EB 2006-09-27
   Key fingerprint = 3534 10A3 F143 B760 EF3E BEDF 8509 6A06 AEB9 10EB
uid          Kai Wang <kaiw@FreeBSD.org>
uid          Kai Wang <kaiw@student.chalmers.se>
uid          Kai Wang <kaiwang27@gmail.com>
uid          Kai Wang <kaiw27@gmail.com>
sub 2048g/1D5AA4DD 2006-09-27
```

D.3.352. Adam Weinberger <adamw@FreeBSD.org>

```
pub 1024D/42C743FD 2002-10-12 Adam Weinberger <adam@vectors.cx>
   Key fingerprint = A980 3F2E 80A8 9619 9D1C 82E8 A3C2 8CD9 42C7 43FD
sub 1024g/15D67628 2002-10-12
```

D.3.353. Peter Wemm <peter@FreeBSD.org>

```
pub 1024D/7277717F 2003-12-14 Peter Wemm <peter@wemm.org>
   Key fingerprint = 622B 2282 E92B 3BAB 57D1 A417 1512 AE52 7277 717F
uid          Peter Wemm <peter@FreeBSD.ORG>
sub 1024g/8B40D9D1 2003-12-14
pub 1024R/D89CE319 1995-04-02 Peter Wemm <peter@netplex.com.au>
   Key fingerprint = 47 05 04 CA 4C EE F8 93 F6 DB 02 92 6D F5 58 8A
uid          Peter Wemm <peter@perth.dialix.oz.au>
uid          Peter Wemm <peter@haywire.dialix.com>
```

D.3.354. Nathan Whitehorn <nwhitehorn@FreeBSD.org>

```
pub 1024D/FC118258 2008-07-03
    Key fingerprint = A399 BEA0 8D2B 63B3 47B5 056D 8513 5B96 FC11 8258
uid      Nathan Whitehorn <nwhitehorn@freebsd.org>
uid      Nathan Whitehorn <nwhitehorn@icecube.wisc.edu>
uid      Nathan Whitehorn <nwhitehorn@physics.wisc.edu>
uid      Nathan Whitehorn <whitehorn@wisc.edu>
sub 2048g/EDB55363 2008-07-03
```

D.3.355. Martin Wilke <miwi@FreeBSD.org>

```
pub 1024D/B1E6FCE9 2009-01-31
    Key fingerprint = C022 7D60 F598 8188 2635 0F6E 74B2 4884 B1E6 FCE9
uid      Martin Wilke <miwi@FreeBSD.org>
sub 4096g/096DA69D 2009-01-31
```

D.3.356. Nate Williams <nate@FreeBSD.org>

```
pub 1024D/C2AC6BA4 2002-01-28 Nate Williams (FreeBSD) <nate@FreeBSD.org>
    Key fingerprint = 8EE8 5E72 8A94 51FA EA68 E001 FFF9 8AA9 C2AC 6BA4
sub 1024g/03EE46D2 2002-01-28
```

D.3.357. Steve Wills <swills@FreeBSD.org>

```
pub 2048R/207B1BA1 2010-09-02 [expires: 2011-09-02]
    Key fingerprint = 98FA 414A 5C2A 0EF9 CFD0 AD0D F5CF 62B3 207B 1BA1
uid      Steve Wills <swills@freebsd.org>
uid      Steve Wills <steve@mouf.net>
sub 2048R/E9B254FD 2010-09-02 [expires: 2011-09-02]
```

D.3.358. Thomas Wintergerst <twinterg@FreeBSD.org>

```
pub 1024D/C45CB978 2006-01-08
    Key fingerprint = 04EE 8114 7C6D 22CE CDC8 D7F8 112D 01DB C45C B978
uid      Thomas Wintergerst <twinterg@gmx.de>
uid      Thomas Wintergerst <twinterg@freebsd.org>
uid      Thomas Wintergerst
uid      Thomas Wintergerst <thomas.wintergerst@nord-com.net>
uid      Thomas Wintergerst <thomas.wintergerst@materna.de>
sub 2048g/3BEBEF8A 2006-01-08
sub 1024D/8F631374 2006-01-08
sub 2048g/34F631DC 2006-01-08
```

D.3.359. Garrett Wollman <wollman@FreeBSD.org>

```
pub 1024D/0B92FAEA 2000-01-20 Garrett Wollman <wollman@FreeBSD.org>
   Key fingerprint = 4627 19AF 4649 31BF DE2E 3C66 3ECF 741B 0B92 FAEA
sub 1024g/90D5EBC2 2000-01-20
```

D.3.360. Jörg Wunsch <joerg@FreeBSD.org>

```
pub 1024D/69A85873 2001-12-11 Joerg Wunsch <j@uriah.heep.sax.de>
   Key fingerprint = 5E84 F980 C3CA FD4B B584 1070 F48C A81B 69A8 5873
pub 1024D/69A85873 2001-12-11 Joerg Wunsch <j@uriah.heep.sax.de>
uid      Joerg Wunsch <joerg_wunsch@interface-systems.de>
uid      Joerg Wunsch <joerg@FreeBSD.org>
uid      Joerg Wunsch <j@ida.interface-business.de>
sub 1024g/21DC9924 2001-12-11
```

D.3.361. David Xu <davidxu@FreeBSD.org>

```
pub 1024D/48F2BDAB 2006-07-13 [expires: 2009-07-12]
   Key fingerprint = 7182 434F 8809 A4AF 9AE8 F1B5 12F6 3390 48F2 BDAB
uid      David Xu <davidxu@freebsd.org>
sub 4096g/ED7DB38A 2006-07-13 [expires: 2009-07-12]
```

D.3.362. Maksim Yevmenkin <emax@FreeBSD.org>

```
pub 1024D/F050D2DD 2003-10-01 Maksim Yevmenkin <m_evmenkin@yahoo.com>
   Key fingerprint = 8F3F D359 E318 5641 8C81 34AD 791D 53F5 F050 D2DD
```

D.3.363. Bjoern A. Zeeb <bz@FreeBSD.org>

```
pub 1024D/3CCF1842 2007-02-20
   Key fingerprint = 1400 3F19 8FEF A3E7 7207 EE8D 2B58 B8F8 3CCF 1842
uid      Bjoern A. Zeeb <bz@zabbadoz.net>
uid      Bjoern A. Zeeb <bzeeb@zabbadoz.net>
uid      Bjoern A. Zeeb <bz@FreeBSD.org>
uid      Bjoern A. Zeeb <bzeeb-lists@lists.zabbadoz.net>
sub 4096g/F36BDC5D 2007-02-20
```

D.3.364. Alexey Zelkin <phantom@FreeBSD.org>

```
pub 1024D/9196B7D9 2002-01-28 Alexey Zelkin <phantom@FreeBSD.org>
   Key fingerprint = 4465 F2A4 28C1 C2E4 BB95 1EA0 C70D 4964 9196 B7D9
sub 1024g/E590ABA4 2002-01-28
```

D.3.365. Sepherosa Ziehau <sephe@FreeBSD.org>

```
pub 2048R/3E51FB42 2005-10-21
    Key fingerprint = 5F47 3861 7ABA 8773 9E32 0474 5C33 841C 3E51 FB42
uid      Sepherosa Ziehau (freebsd) <sephe@freebsd.org>
uid      Sepherosa Ziehau (sephe) <sepherosa@gmail.com>
sub 2048R/7AA31321 2005-10-21
```

FreeBSD Glossary

This glossary contains terms and acronyms used within the FreeBSD community and documentation.

A

ACL

Patrz: Access Control List

ACPI

Patrz: Advanced Configuration and Power Interface

AMD

Patrz: Automatic Mount Daemon

AML

Patrz: ACPI Machine Language

API

Patrz: Application Programming Interface

APIC

Patrz: Advanced Programmable Interrupt Controller

APM

Patrz: Advanced Power Management

APOP

Patrz: Authenticated Post Office Protocol

ASL

Patrz: ACPI Source Language

ATA

Patrz: Advanced Technology Attachment

ATM

Patrz: Asynchronous Transfer Mode

ACPI Machine Language

Pseudocode, interpreted by a virtual machine within an ACPI-compliant operating system, providing a layer between the underlying hardware and the documented interface presented to the OS.

ACPI Source Language

The programming language AML is written in.

Access Control List

A list of permissions attached to an object, usually either a file or a network device.

Advanced Configuration and Power Interface

A specification which provides an abstraction of the interface the hardware presents to the operating system, so that the operating system should need to know nothing about the underlying hardware to make the most of it. ACPI evolves and supersedes the functionality provided previously by APM, PNPBIOS and other technologies, and provides facilities for controlling power consumption, machine suspension, device enabling and disabling, etc.

Application Programming Interface

A set of procedures, protocols and tools that specify the canonical interaction of one or more program parts; how, when and why they do work together, and what data they share or operate on.

Advanced Power Management

An API enabling the operating system to work in conjunction with the BIOS in order to achieve power management. APM has been superseded by the much more generic and powerful ACPI specification for most applications.

Advanced Programmable Interrupt Controller

Advanced Technology Attachment

Asynchronous Transfer Mode

Authenticated Post Office Protocol

Automatic Mount Daemon

A daemon that automatically mounts a filesystem when a file or directory within that filesystem is accessed.

B

BAR

Patrz.: Base Address Register

BIND

Patrz.: Berkeley Internet Name Domain

BIOS

Patrz.: Basic Input/Output System

BSD

Patrz.: Berkeley Software Distribution

Base Address Register

The registers that determine which address range a PCI device will respond to.

Basic Input/Output System

The definition of BIOS depends a bit on the context. Some people refer to it as the ROM chip with a basic set of routines to provide an interface between software and hardware. Others refer to it as the set of routines contained in the chip that help in bootstrapping the system. Some might also refer to it as the screen used to configure the bootstrapping process. The BIOS is PC-specific but other systems have something similar.

Berkeley Internet Name Domain

An implementation of the DNS protocols.

Berkeley Software Distribution

This is the name that the Computer Systems Research Group (CSRG) at The University of California at Berkeley (<http://www.berkeley.edu>) gave to their improvements and modifications to AT&T's 32V UNIX. FreeBSD is a descendant of the CSRG work.

Bikeshed Building

A phenomenon whereby many people will give an opinion on an uncomplicated topic, whilst a complex topic receives little or no discussion. See the FAQ (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/books/faq/misc.html#BIKESHED-PAINTING) for the origin of the term.

C

CD

Patrz: Carrier Detect

CHAP

Patrz: Challenge Handshake Authentication Protocol

CLIP

Patrz: Classical IP over ATM

COFF

Patrz: Common Object File Format

CPU

Patrz: Central Processing Unit

CTS

Patrz: Clear To Send

CVS

Patrz: Concurrent Versions System

Carrier Detect

An RS232C signal indicating that a carrier has been detected.

Central Processing Unit

Also known as the processor. This is the brain of the computer where all calculations take place. There are a number of different architectures with different instruction sets. Among the more well-known are the Intel-x86 and derivatives, Sun SPARC, PowerPC, and Alpha.

Challenge Handshake Authentication Protocol

A method of authenticating a user, based on a secret shared between client and server.

Classical IP over ATM

Clear To Send

An RS232C signal giving the remote system permission to send data.

Patrz. też: Request To Send.

Common Object File Format

Concurrent Versions System

A version control system, providing a method of working with and keeping track of many different revisions of files. CVS provides the ability to extract, merge and revert individual changes or sets of changes, and offers the ability to keep track of which changes were made, by who and for what reason.

D

DAC

Patrz: Discretionary Access Control

DDB

Patrz: Debugger

DES

Patrz: Data Encryption Standard

DHCP

Patrz: Dynamic Host Configuration Protocol

DNS

Patrz: Domain Name System

DSDT

Patrz: Differentiated System Description Table

DSR

Patrz: Data Set Ready

DTR

Patrz: Data Terminal Ready

DVMRP

Patrz: Distance-Vector Multicast Routing Protocol

Discretionary Access Control

Data Encryption Standard

A method of encrypting information, traditionally used as the method of encryption for UNIX passwords and the crypt(3) function.

Data Set Ready

An RS232C signal sent from the modem to the computer or terminal indicating a readiness to send and receive data.

Patrz też: Data Terminal Ready.

Data Terminal Ready

An RS232C signal sent from the computer or terminal to the modem indicating a readiness to send and receive data.

Debugger

An interactive in-kernel facility for examining the status of a system, often used after a system has crashed to establish the events surrounding the failure.

Differentiated System Description Table

An ACPI table, supplying basic configuration information about the base system.

Distance-Vector Multicast Routing Protocol

Domain Name System

The system that converts humanly readable hostnames (i.e., mail.example.net) to Internet addresses and vice versa.

Dynamic Host Configuration Protocol

A protocol that dynamically assigns IP addresses to a computer (host) when it requests one from the server. The address assignment is called a „lease”.

E

ECOFF

Patrz: Extended COFF

ELF

Patrz: Executable and Linking Format

ESP

Patrz: Encapsulated Security Payload

Encapsulated Security Payload

Executable and Linking Format

Extended COFF

F

FADT

Patrz: Fixed ACPI Description Table

FAT

Patrz: File Allocation Table

FAT16

Patrz: File Allocation Table (16-bit)

FTP

Patrz: File Transfer Protocol

File Allocation Table

File Allocation Table (16-bit)**File Transfer Protocol**

A member of the family of high-level protocols implemented on top of TCP which can be used to transfer files over a TCP/IP network.

Fixed ACPI Description Table**G****GUI**

Patrz: Graphical User Interface

Giant

The name of a mutual exclusion mechanism (a `sleep mutex`) that protects a large set of kernel resources. Although a simple locking mechanism was adequate in the days where a machine might have only a few dozen processes, one networking card, and certainly only one processor, in current times it is an unacceptable performance bottleneck. FreeBSD developers are actively working to replace it with locks that protect individual resources, which will allow a much greater degree of parallelism for both single-processor and multi-processor machines.

Graphical User Interface

A system where the user and computer interact with graphics.

H**HTML**

Patrz: HyperText Markup Language

HUP

Patrz: HangUp

HangUp

HyperText Markup Language

The markup language used to create web pages.

I**I/O**

Patrz: Input/Output

IASL

Patrz: Intel's ASL compiler

IMAP

Patrz: Internet Message Access Protocol

IP

Patrz: Internet Protocol

IPFW

Patrz: IP Firewall

IPP

Patrz: Internet Printing Protocol

IPv4

Patrz: IP Version 4

IPv6

Patrz: IP Version 6

ISP

Patrz: Internet Service Provider

IP Firewall**IP Version 4**

The IP protocol version 4, which uses 32 bits for addressing. This version is still the most widely used, but it is slowly being replaced with IPv6.

Patrz, też: IP Version 6.

IP Version 6

The new IP protocol. Invented because the address space in IPv4 is running out. Uses 128 bits for addressing.

Input/Output

Intel's ASL compiler

Intel's compiler for converting ASL into AML.

Internet Message Access Protocol

A protocol for accessing email messages on a mail server, characterised by the messages usually being kept on the server as opposed to being downloaded to the mail reader client.

Patrz też: Post Office Protocol Version 3.

Internet Printing Protocol

Internet Protocol

The packet transmitting protocol that is the basic protocol on the Internet. Originally developed at the U.S. Department of Defense and an extremely important part of the TCP/IP stack. Without the Internet Protocol, the Internet would not have become what it is today. For more information, see RFC 791 (<ftp://ftp.rfc-editor.org/in-notes/rfc791.txt>).

Internet Service Provider

A company that provides access to the Internet.

K

KAME

Japanese for „turtle”, the term KAME is used in computing circles to refer to the KAME Project (<http://www.kame.net/>), who work on an implementation of IPv6.

KDC

Patrz: Key Distribution Center

KLD

Patrz: Kernel ld(1)

KSE

Patrz: Kernel Scheduler Entities

KVA

Patrz: Kernel Virtual Address

Kbps

Patrz: Kilo Bits Per Second

Kernel ld(1)

A method of dynamically loading functionality into a FreeBSD kernel without rebooting the system.

Kernel Scheduler Entities

A kernel-supported threading system. See the project home page (<http://www.FreeBSD.org/kse>) for further details.

Kernel Virtual Address

Key Distribution Center

Kilo Bits Per Second

Used to measure bandwidth (how much data can pass a given point at a specified amount of time). Alternates to the Kilo prefix include Mega, Giga, Tera, and so forth.

L

LAN

Patrz: Local Area Network

LOR

Patrz: Lock Order Reversal

LPD

Patrz: Line Printer Daemon

Line Printer Daemon**Local Area Network**

A network used on a local area, e.g. office, home, or so forth.

Lock Order Reversal

The FreeBSD kernel uses a number of resource locks to arbitrate contention for those resources. A run-time lock diagnostic system found in FreeBSD-CURRENT kernels (but removed for releases), called witness(4), detects the potential for deadlocks due to locking errors. (witness(4) is actually slightly conservative, so it is possible to get false positives.) A true positive report indicates that „if you were unlucky, a deadlock would have happened here”.

True positive LORs tend to get fixed quickly, so check <http://lists.FreeBSD.org/mailman/listinfo/freebsd-current> and the LORs Seen (<http://sources.zabbadoz.net/freebsd/lor.html>) page before posting to the mailing lists.

M**MAC**

Patrz: Mandatory Access Control

MADT

Patrz: Multiple APIC Description Table

MFC

Patrz: Merge From Current

MFP4

Patrz: Merge From Perforce

MFS

Patrz: Merge From Stable

MIT

Patrz: Massachusetts Institute of Technology

MLS

Patrz: Multi-Level Security

MOTD

Patrz: Message Of The Day

MTA

Patrz: Mail Transfer Agent

MUA

Patrz: Mail User Agent

Mail Transfer Agent

An application used to transfer email. An MTA has traditionally been part of the BSD base system. Today Sendmail is included in the base system, but there are many other MTAs, such as postfix, qmail and Exim.

Mail User Agent

An application used by users to display and write email.

Mandatory Access Control

Massachusetts Institute of Technology

Merge From Current

To merge functionality or a patch from the -CURRENT branch to another, most often -STABLE.

Merge From Perforce

To merge functionality or a patch from the Perforce repository to the -CURRENT branch.

Patrz też: Perforce.

Merge From Stable

In the normal course of FreeBSD development, a change will be committed to the -CURRENT branch for testing before being merged to -STABLE. On rare occasions, a change will go into -STABLE first and then be merged to -CURRENT.

This term is also used when a patch is merged from -STABLE to a security branch.

Patrz też: Merge From Current.

Message Of The Day

A message, usually shown on login, often used to distribute information to users of the system.

Multi-Level Security

Multiple APIC Description Table

N

NAT

Patrz: Network Address Translation

NDISulator

Patrz: Project Evil

NFS

Patrz: Network File System

NTFS

Patrz: New Technology File System

NTP

Patrz: Network Time Protocol

Network Address Translation

A technique where IP packets are rewritten on the way through a gateway, enabling many machines behind the gateway to effectively share a single IP address.

Network File System

New Technology File System

A filesystem developed by Microsoft and available in its „New Technology” operating systems, such as Windows 2000, Windows NT and Windows XP.

Network Time Protocol

A means of synchronizing clocks over a network.

O

OBE

Patrz: Overtaken By Events

ODMR

Patrz: On-Demand Mail Relay

OS

Patrz: Operating System

On-Demand Mail Relay

Operating System

A set of programs, libraries and tools that provide access to the hardware resources of a computer. Operating systems range today from simplistic designs that support only one program running at a time, accessing only one device to fully multi-user, multi-tasking and multi-process systems that can serve thousands of users simultaneously, each of them running dozens of different applications.

Overtaken By Events

Indicates a suggested change (such as a Problem Report or a feature request) which is no longer relevant or applicable due to such things as later changes to FreeBSD, changes in networking standards, the affected hardware having since become obsolete, and so forth.

P

p4

Patrz: Perforce

PAE

Patrz: Physical Address Extensions

PAM

Patrz: Pluggable Authentication Modules

PAP

Patrz: Password Authentication Protocol

PC

Patrz: Personal Computer

PCNSFD

Patrz: Personal Computer Network File System Daemon

PDF

Patrz: Portable Document Format

PID

Patrz: Process ID

POLA

Patrz: Principle Of Least Astonishment

POP

Patrz: Post Office Protocol

POP3

Patrz: Post Office Protocol Version 3

PPD

Patrz: PostScript Printer Description

PPP

Patrz: Point-to-Point Protocol

PPPoA

Patrz: PPP over ATM

PPPoE

Patrz: PPP over Ethernet

PPP over ATM

PPP over Ethernet

PR

Patrz: Problem Report

PXE

Patrz: Preboot eXecution Environment

Password Authentication Protocol

Perforce

A source code control product made by Perforce Software (<http://www.perforce.com/>) which is more advanced than CVS. Although not open source, its use is free of charge to open-source projects such as FreeBSD.

Some FreeBSD developers use a Perforce repository as a staging area for code that is considered too experimental for the -CURRENT branch.

Personal Computer

Personal Computer Network File System Daemon

Physical Address Extensions

A method of enabling access to up to 64 GB of RAM on systems which only physically have a 32-bit wide address space (and would therefore be limited to 4 GB without PAE).

Pluggable Authentication Modules

Point-to-Point Protocol

Pointy Hat

A mythical piece of headgear, much like a dunce cap, awarded to any FreeBSD committer who breaks the build, makes revision numbers go backwards, or creates any other kind of havoc in the source base. Any committer worth his or her salt will soon accumulate a large collection. The usage is (almost always?) humorous.

Portable Document Format

Post Office Protocol

Patrz też: Post Office Protocol Version 3.

Post Office Protocol Version 3

A protocol for accessing email messages on a mail server, characterised by the messages usually being downloaded from the server to the client, as opposed to remaining on the server.

Patrz też: Internet Message Access Protocol.

PostScript Printer Description

Preboot eXecution Environment

Principle Of Least Astonishment

As FreeBSD evolves, changes visible to the user should be kept as unsurprising as possible. For example, arbitrarily rearranging system startup variables in `/etc/defaults/rc.conf` violates POLA. Developers consider POLA when contemplating user-visible system changes.

Problem Report

A description of some kind of problem that has been found in either the FreeBSD source or documentation. See [Writing FreeBSD Problem Reports](http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/problem-reports/index.html) (http://www.FreeBSD.org/doc/pl_PL.ISO8859-2/articles/problem-reports/index.html).

Process ID

A number, unique to a particular process on a system, which identifies it and allows actions to be taken against it.

Project Evil

The working title for the NDISulator, written by Bill Paul, who named it referring to how awful it is (from a philosophical standpoint) to need to have something like this in the first place. The NDISulator is a special compatibility module to allow Microsoft Windows™ NDIS miniport network drivers to be used with FreeBSD/i386. This is usually the only way to use cards where the driver is closed-source. See `src/sys/compat/ndis/subr_ndis.c`.

R

RA

Patrz: Router Advertisement

RAID

Patrz: Redundant Array of Inexpensive Disks

RAM

Patrz: Random Access Memory

RD

Patrz: Received Data

RFC

Patrz: Request For Comments

RISC

Patrz: Reduced Instruction Set Computer

RPC

Patrz: Remote Procedure Call

RS232C

Patrz: Recommended Standard 232C

RTS

Patrz: Request To Send

Random Access Memory**Revision Control System**

The *Revision Control System* (RCS) is one of the oldest software suites that implement „revision control” for plain files. It allows the storage, retrieval, archival, logging, identification and merging of multiple revisions for each file. RCS consists of many small tools that work together. It lacks some of the features found in more modern revision control systems, like CVS or Subversion, but it is very simple to install, configure, and start using for a small set of files. Implementations of RCS can be found on every major UNIX-like OS.

Patrz, też: Concurrent Versions System, Subversion.

Received Data

An RS232C pin or wire that data is received on.

Patrz, też: Transmitted Data.

Recommended Standard 232C

A standard for communications between serial devices.

Reduced Instruction Set Computer

An approach to processor design where the operations the hardware can perform are simplified but made as general purpose as possible. This can lead to lower power consumption, fewer transistors and in some cases,

better performance and increased code density. Examples of RISC processors include the Alpha, SPARC, ARM and PowerPC.

Redundant Array of Inexpensive Disks

Remote Procedure Call

repocopy

Patrz: Repository Copy

Repository Copy

A direct copying of files within the CVS repository.

Without a repocopy, if a file needed to be copied or moved to another place in the repository, the committer would run `cvsv add` to put the file in its new location, and then `cvsv rm` on the old file if the old copy was being removed.

The disadvantage of this method is that the history (i.e. the entries in the CVS logs) of the file would not be copied to the new location. As the FreeBSD Project considers this history very useful, a repository copy is often used instead. This is a process where one of the repository masters will copy the files directly within the repository, rather than using the `cvsv(1)` program.

Request For Comments

A set of documents defining Internet standards, protocols, and so forth. See www.rfc-editor.org (<http://www.rfc-editor.org/>).

Also used as a general term when someone has a suggested change and wants feedback.

Request To Send

An RS232C signal requesting that the remote system commence transmission of data.

Patrz. też: Clear To Send.

Router Advertisement

S

SCI

Patrz: System Control Interrupt

SCSI

Patrz: Small Computer System Interface

SG

Patrz: Signal Ground

SMB

Patrz: Server Message Block

SMP

Patrz: Symmetric MultiProcessor

SMTP

Patrz: Simple Mail Transfer Protocol

SMTP AUTH

Patrz: SMTP Authentication

SSH

Patrz: Secure Shell

STR

Patrz: Suspend To RAM

SVN

Patrz: Subversion

SMTP Authentication

Server Message Block

Signal Ground

An RS232 pin or wire that is the ground reference for the signal.

Simple Mail Transfer Protocol

Secure Shell

Small Computer System Interface

Subversion

Subversion is a version control system, similar to CVS, but with an expanded feature list.

Patrz. też: Concurrent Versions System.

Suspend To RAM

Symmetric MultiProcessor

System Control Interrupt

T

TCP

Patrz: Transmission Control Protocol

TCP/IP

Patrz: Transmission Control Protocol/Internet Protocol

TD

Patrz: Transmitted Data

TFTP

Patrz: Trivial FTP

TGT

Patrz: Ticket-Granting Ticket

TSC

Patrz: Time Stamp Counter

Ticket-Granting Ticket

Time Stamp Counter

A profiling counter internal to modern Pentium processors that counts core frequency clock ticks.

Transmission Control Protocol

A protocol that sits on top of (e.g.) the IP protocol and guarantees that packets are delivered in a reliable, ordered, fashion.

Transmission Control Protocol/Internet Protocol

The term for the combination of the TCP protocol running over the IP protocol. Much of the Internet runs over TCP/IP.

Transmitted Data

An RS232C pin or wire that data is transmitted on.

Patrz. też: Received Data.

Trivial FTP

U

UDP

Patrz. User Datagram Protocol

UFS1

Patrz. Unix File System Version 1

UFS2

Patrz. Unix File System Version 2

UID

Patrz. User ID

URL

Patrz. Uniform Resource Locator

USB

Patrz: Universal Serial Bus

Uniform Resource Locator

A method of locating a resource, such as a document on the Internet and a means to identify that resource.

Unix File System Version 1

The original UNIX file system, sometimes called the Berkeley Fast File System.

Unix File System Version 2

An extension to UFS1, introduced in FreeBSD 5-CURRENT. UFS2 adds 64 bit block pointers (breaking the 1T barrier), support for extended file storage and other features.

Universal Serial Bus

A hardware standard used to connect a wide variety of computer peripherals to a universal interface.

User ID

A unique number assigned to each user of a computer, by which the resources and permissions assigned to that user can be identified.

User Datagram Protocol

A simple, unreliable datagram protocol which is used for exchanging data on a TCP/IP network. UDP does not provide error checking and correction like TCP.

V

VPN

Patrz: Virtual Private Network

Virtual Private Network

A method of using a public telecommunication such as the Internet, to provide remote access to a localized network, such as a corporate LAN.

Colophon

Niniejsza książka jest dziełem setek osób z „Projektu Dokumentacji FreeBSD”. Tekst jest przygotowywany w języku SGML zgodnie ze standardem DocBook DTD, a następnie konwertowany do całej rzeszy innych formatów za pomocą modułu DSSSL **Jade**. Instrukcje formatowania tekstu zostały przygotowane przy wykorzystaniu arkuszy styli DSSSL Norma Walsh. Przygotowanie wersji do wydruku nie byłoby możliwe gdyby nie język składu tekstu T_EX Donalda Knutha, L^AT_EX Lesliego Lamporta, czy makra **JadeT_EX** Sebastiana Rahtza.